Tilburg centre for Cognition and Communication
Tilburg University
http://www.uvt.nl/ticc

P.O. Box 90153
5000 LE Tilburg, The Netherlands
Email: ticc@uvt.nl

March 14, 2013

TiCC TR 2013–001

A methodology for designing semantic annotations

Harry Bunt
TiCC, Tilburg University

Abstract

This paper presents a methodology for designing languages for semantic annotation. Central in this methodology is the specification of representation formats as renderings of conceptual structures defined by an *abstract syntax* as set-theoretic constructs. An *ideal* representation format is defined as one that is able to represent all the conceptual distinctions made in the abstract syntax, and of which each representation encodes one and only one structure defined by the abstract syntax. The semantics of an annotation language is defined for its *abstract* syntax and is shared by all its representation formats; every ideal representation format is therefore convertible through a meaning-preserving mapping to any other ideal representation format. The methodology is called CASCADES after its four stages: **C**onceptual analysis, **A**bstract syntax, **S**emantics and **C**oncrete syntax for **A**nnotation **DES**ign.
The CASCADES model derives its usefulness from supporting a systematic design process for semantic annotations, giving due attention to the conceptual and semantic issues underlying choices in annotation formats, including support in the form of procedures for how to move from one stage of the design process to another, in particular for how to construct an abstract syntax given a conceptual analysis; how to define a semantics for a given abstract syntax; and how to map an abstract syntax to an XML-based concrete syntax.
Three applications of the CASCADES methodology are discussed: (1) the design of an ISO standard for dialogue annotation starting from a conceptual analysis; (2) the analysis of existing annotation schemes such as those of the Penn Discourse Treebank and TimeML, as a basis for the development of ISO standards for semantic annotation; (3) the detection and repair of deficiencies in existing annotation schemes.

1 Introduction

The linguistic annotation of a text often takes the form of attaching certain labels, such as part of speech labels or named entity tags, to textual elements. In the case of semantic annotation, however, the information to be captured by annotations typically has a complexity which requires more than simple labeling, and instead uses expressions in an annotation language. For example, Pustejovsky et al. (2003; 2005; 2007) developed the language TimeML for the annotation of texts with information relating to time and events.

An annotation language being a formal language, one might expect its definition to follow the usual setup of formal language definitions and to consist of a syntax, which defines a class of well-formed expressions, and a semantics, which specifies what these expressions mean. The definition of XML, which is the basis of most current representation formats used for annotation, is an exception in this respect, since it consists of a syntax only; the interpretation of its expressions is left to the user (human or machine). This creates a fundamental problem for applying XML in semantic annotation: semantic annotations are meant to capture part of the meaning of the annotated text, but if the annotations don't have a well-defined meaning, then why would they capture meaning better than the text itself? Bunt & Romary (2002) therefore formulated the requirement of *semantic adequacy* for a semantic annotation language, which says that such a language should have a well-defined semantics.

The Linguistic Annotation Framework (Ide & Romary, 2004; ISO 24612:2012) 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, while the term 'representation' refers to the format in which an annotation is rendered, independent of its content. According to the Linguistic Annotation Framework, annotations are the proper level of standardisation, rather than representations.

This paper proposes an approach to the definition of annotation languages which meets both the requirement of semantic adequacy and the requirements of the Linguistic Annotation Framework. According to this approach, the definition of an annotation language has a syntactic component that specifies, besides a class of *representation structures* also a class of more abstract *annotation structures*. These two parts are called the *concrete* and *abstract syntax* of the language definition, respectively, and their distinction is a reflection of the distinction between annotations and representations. Moreover, a semantics is defined for the *abstract* rather than for the concrete syntax, and is inherited by the representations defined by a concrete syntax. The approach distinguishes four stages in the process of designing a semantic annotation language: (1) conceptual analysis of the information to be captured by annotations; (2) specification of an abstract syntax defining the basic concepts in annotations and their possible combinations; (3) definition of a semantics for the annotation structures defined by the abstract syntax; and (4) specification of a concrete syntax for representing the annotation structures defined by the abstract syntax. Because of these stages, the approach is called CASCADES (**C**onceptual analysis, **A**bstract syntax, **S**emantics, and **C**oncrete syntax for **A**nnotation **DES**ign).

This approach was developed during the ISO project *Semantic annotation framework, Part 1: Time and events* (ISO 24617-1, 2012), and incorporates ideas that have evolved over the years in annotation research, such as the specification of a metamodel in an early stage of ISO standardisation projects (see e.g. Bunt & Romary, 2004); the distinction between annotations and representations in the Linguistic Annotation Framework (Ide & Romary, 2004; 2006); the use of standoff rather than embedded annotation; and the definition of general pivot formats like the graphic format GrAF (Ide & Suderman, 2007). New in the CASCADES approach is that these ideas are incorporated in a structured design process with outlines of procedures for moving from one stage of design to another (in either direction), in combination with the new ideas of an abstract syntax for annotations; a semantics for an abstract syntax, and the notion of 'ideal representation formats' for a given abstract syntax.

The CASCADES methodology has three kinds of applications, which are discussed in this paper. First, the methodology was originally developed to support the definition of interoperable semantic annotations, notably as targeted in ISO annotation standards. Besides its use in the ISO project defining the standard ISO 24617-1 for time and events, including a metamodel and the annotation language ISO-TimeML (see Pustejovsky et al., 2010), it has been used from the start in ISO project *Semantic* *annotation framework, Part 2: Dialogue acts*, beginning with the stage of conceptual analysis and ending with the definition of an XML-based representation format for dialogue annotation in the dialogue act markup language DiAML (see Bunt et al., 2010; 2012).

A second kind of application is to design semantic annotations not starting with a conceptual analysis, but starting from an existing representation format, such as that of PropBank (Palmer et al., 2005) or that of the Penn Discourse Treebank (Prasad et al., 2008), and to use the CASCADES method in reverse direction to 'reverse-engineer' an abstract syntax underlying the representation format. If the aim is to develop an ISO standard for a certain domain of semantic annotation, then the requirement of semantic adequacy implies that a semantics should also be defined, for which the CASCADES procedure can be helpful. The methodology is currently being used for this kind of application in developing ISO standard 24617-4 for the annotation of semantic roles (see Bonial et al., 2011a; 2011b), and ISO 24617-8 for the annotation of discourse relations (see Bunt, Prasad & Joshi, 2012); the latter is discussed in Section 4.1.

A third kind of application is to use the methodology, in particular its feedback cycles, described in Section 3, for detecting and repairing deficiencies in existing annotation schemes and representations. Ide et al. (2011) already noted that the 'reverse-engineering' of an abstract syntax for the representations of the Penn Discourse Treebank lead to considerable improvements in the representations; adding a semantics to the abstract syntax leads to further improvements and ensures that the requirement of semantic adequacy is satisfied; (see Section 4.1). Two other cases of this kind of application are the representation of repeated events in ISO-TimeML and the annotation of dependency relations between dialogue acts in DiAML; these are discussed in Section 4.2.

Section 2 first describes and illustrates the four CASCADES stages. Section 3 outlines procedures for defining an abstract syntax from a given conceptual analysis, for defining an XML-based representation format given the specification of an abstract syntax, and for defining a DRT-based semantics for a given abstract syntax. After the discussion of applications in Section 4, the paper ends with a summary of the innovations and applications of the CASCADES method in Section 5.

2 Four stages in the design of semantic annotations

An *annotation language* serves to represent the information that annotations add to primary data. When designing an annotation language, the first question to consider is what information we want to be able to represent. The design of an annotation language therefore starts at a conceptual level, independent of representation formats.

2.1 Stage 1: Conceptual analysis

A first phase of *conceptual analysis* serves to determine the conceptual content of the targeted annotations, identifying the basic concepts that form the building blocks of the annotations, and establishing the ways in which these concepts are interrelated. This early stage of designing an annotation language results in the establishment of what in ISO projects is called a *metamodel*, a diagrammatic representation of the kinds of elements that may occur in annotations and how they are related. Figure 1 shows an example of a metamodel for dialogue act annotation.

This metamodel says that a dialogue consists of at least two functional segments, as indicated by '2..N' at the head of the arrow relating the 'dialogue' concept and the 'functional segment' concept. A functional segment is related to one or more dialogue acts (more than one in the case of a multifunctional segment). A dialogue act has one sender, one or more addressees, and possibly other participants. It has a semantic content belonging to a certain content dimension and a communicative function, which may have one or more qualifiers (Petukhova & Bunt, 2010). It may also be related to other dialogue acts and to other functional segments through functional dependence relations, rhetorical relations, and feedback dependence relations.

Except for the dialogue concept, which is a kind of meta-concept for dialogue analysis, each of the boxes in the metamodel represents a category of concepts; a labelled arrow between boxes represents a single relational concept rather than a category of relations.

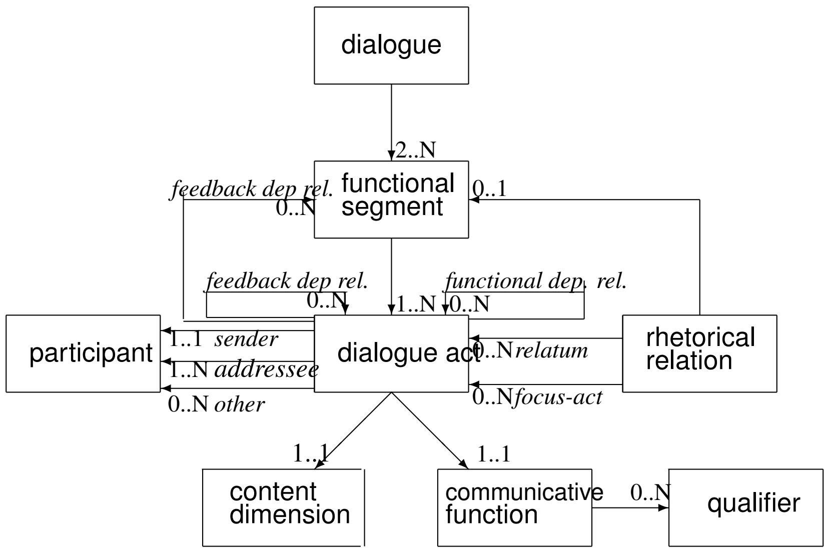

Figure 1: ISO 24617-2 metamodel for dialogue act annotation.

2.2 Stage 2: Specification of an abstract syntax

A second stage in defining a semantic annotation language is the establishment of a formal specification of the concepts in the conceptual analysis, a so-called ‘conceptual inventory’, and of the well-formed combinations of concepts of the conceptual inventory. These specifications form the so-called *abstract syntax* of the annotation language.

The abstract syntax provides a format-independent level of specifying the possible structures of annotations as set-theoretic constructs, called *annotation structures*.An annotation structure is a set of two kinds of elements, *entity structures* and *link structures*. An entity structure contains information about a segment of the primary data (typically a stretch of text, speech, or nonverbal or multimodal communicative behaviour); a link structure contains information about the way in which two or more segments of primary data are semantically related. An entity structure is formally a pair

(1) $\varepsilon = \langle m, s \rangle$

consisting of a markable *m*, identifying a segment of primary data, and the semantic information *s* associated with this segment. The structure of the component *s* depends on the kind of semantic information to be annotated; see e.g. (9) - (11) for the structuring of this component in the case of dialogue act annotation.

A link structure is a triple

(2) $L = \langle \varepsilon, E, \rho \rangle$

consisting of an entity structure ε, a set E of entity structures that ε is related to, and a relational component ρ which is either (a) a pair $\langle R, q \rangle$ formed by a relation R and a (possibly empty) set q of qualifiers that further specify how ε and E are related by R; or else (b) a triple $\langle m, R, q \rangle$ with R and q as before, and m a markable which identifies a segment of primary data that expresses the relation R.

For instance, in semantic role annotation a link structure is a triple $\langle \varepsilon_1, \{\varepsilon_2\}, \langle R_i, \langle\rangle \rangle \rangle$, where ε_1 is an entity structure that contains information about an event, ε_2 is an entity structure that contains information about a participant in the event, and R_i is the semantic role of the participant in the event.

2.3 Stage 3: Semantics definition

While in the second stage nothing is said about the representation of annotation structures, it is clear what the annotation structures are intended to mean. The specification of what an annotation structure means is the specification of a *semantics* for these structures. This is the crucial stage 3 (see Fig. 3) of the method. Any representation of an annotation structure inherits its semantics from the annotation structure that it represents.

Defining the semantics of annotation representations indirectly, via the underlying annotation structures, has the important advantage that any format for representing annotation structures inherits *the same* semantics, which is highly beneficial for improving the interoperability of semantic annotations.

In Section 2.2.4 we will see how a formal semantics can be systematically constructed for a given abstract syntax.

2.4 Stage 4: Specification of a concrete syntax

The fourth and final stage of the CASCADES method is the definition of a reference format for representing the annotation structures defined by the abstract syntax, for example as a linearisation in XML.

A representation format for annotation structures should ideally give an exact expression of the information in annotation structures. A concrete syntax, defining a representation format for a given abstract syntax, is said to be *ideal* if it has the following properties:

(3)
- *Completeness:* every annotation structure defined by the abstract syntax can be represented by an expression defined by the concrete syntax ;
- *Unambiguity:* every representation defined by the concrete syntax is the rendering of one and only one annotation structure defined by the abstract syntax.

Due to its completeness, every ideal concrete syntax RF_i defines a function F_i from annotation structures to RF_i-representations, and due to its 'unambiguity' there is also an inverse function F_i^{-1} from RF_i-representations to annotation structures. Since this holds for *any* ideal concrete syntax, it follows that any two ideal representation formats are semantically equivalent.

If I_a is the interpretation function defining the semantics of the abstract syntax, then the meaning $\mu(r)$ of a representation r in some ideal format RF_i is defined by (4).

(4) $\mu(r) =_D I_a(F_i^{-1}(r))$

Figure 2 visualises the relations between abstract syntax, semantics, and multiple ideal concrete syntactic specifications. It is immediately clear that a given representation r defined by concrete syntax RF_i can be converted into a semantically equivalent representation r' in the representation format FR_j by first applying the function F_i^{-1} in order to determine the annotation structure which it encodes, and applying to that annotation structure the function F_j which provides an encoding in the format RF_j.[1]

In other words, for any two ideal representation formats RF_i and RF_j there is a meaning-preserving conversion C_{ij} from RF_i-representations to RF_j-representations defined by function composition:

(5) $C_{ij}(r) =_D F_j \circ F_i^{-1}$

This mapping is meaning-preserving due to the fact that the meaning of a representation is the meaning of the annotation structure that it encodes.

In sum, the CASCADES approach to designing annotation languages consists of the following stages:

[1]The conditions (3) defining an ideal concrete syntax require every annotation structure to have *at least* one representation, but do not rule out the possibility that an annotation structure has more than one representation. The encoding functions, like F_j, may therefore assign multiple RF_j-representations to a given annotation structure.

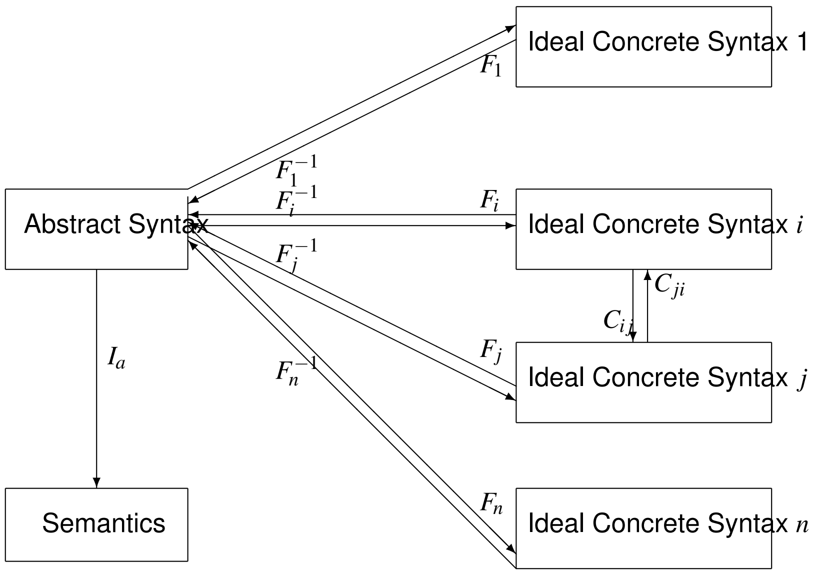

Figure 2: Relations among components of annotation language definition and ideal representation formats.

(6) 1: **Conceptual Analysis**: Formulate a conceptual view of the information to be captured in annotations. Such a formulation is initially informal, in the form of a formulated in natural language. This analysis is conveniently cast in the form of a 'metamodel', i.e. a listing of the categories of entities and relations that play a role in the annotation, visualised in a diagram.

2: **Abstract Syntax**: Articulate the conceptual view in the form of an inventory of basic concepts (a 'conceptual inventory'), and a formal specification of the possible ways of combining these elements in set-theoretical structures, called *annotation structures*. This specification is called an 'abstract syntax'.

3: **Semantics**: Provide a formal semantics for the structures defined by the abstract syntax. Every concrete syntax which defines a rendering of the abstract annotation structures inherits the semantics of the abstract syntax.

4: **Concrete Syntax**: Specify a representation format for the structures defined by the abstract syntax; in particular, specify a representation format that has the properties of being complete and unambiguous - an ideal representation format.

3 Steps and cycles in the design process

The CASCADES model derives its usefulness in the first place from enabling a systematic design process, in which due attention is given to the conceptual and semantic choices on which more superficial decisions such as the choice of particular XML attributes and values should be based. Second, the model provides methodological support by means of procedures for how to make the step from one level of decision-making to the next, in particular for (1) how to construct an abstract syntax given a metamodel (step 1 in Fig. 3); (2) how to define a formal semantics for a given abstract syntax (step 2); and (3) how to map an abstract syntax to an XML-based concrete syntax. These procedures are outlined in the rest of this section.

While the outlined procedures for making these design steps is potentially of great help for producing systematic and well-founded designs, it would be an illusion to think that fully satisfactory designs can be developed through a simple linear sequence of steps from conceptual analysis to the definition of a representation format. Realistic design processes require feedback loops. Figure 3 shows four such loops. First, the specification of an abstract syntax is a way to formalise the conceptual analysis in the

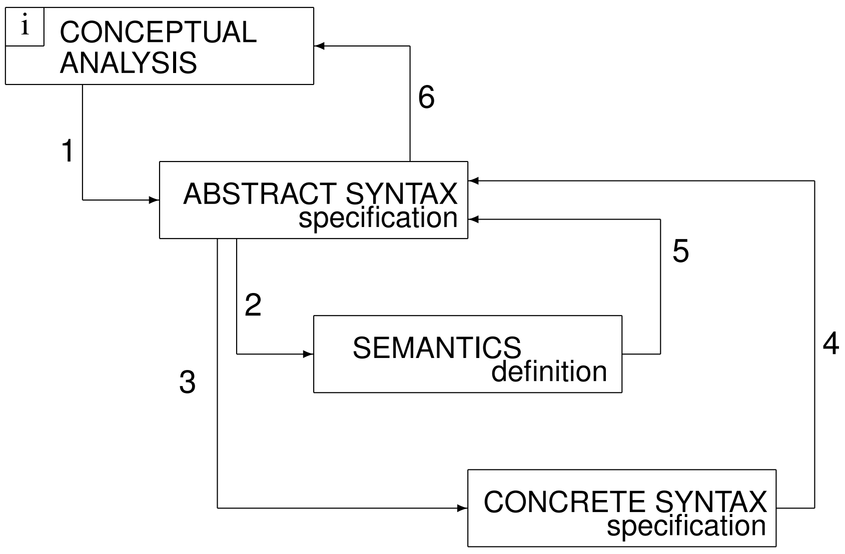

Figure 3: The CASCADES model.

initial stage of the process. This formalisation may very well clarify or alter some aspects of the initial analysis; CASCADES step 6 is for feeding the results of the formalisation back into the conceptual analysis. Second, the specification of a concrete syntax, defining a specific representation format, may by virtue of its concreteness motivate adaptations in the underlying abstract syntax; step 4 is for this feedback in the design process. Third, since the definition of a semantics for an abstract syntax is the best way to find inadequacies in the latter, this may be fed back into the abstract syntax specification in step 5. And finally, the latter two feedback loops may well be combined: if the feedback in step 4 has resulted in a revised specification of the abstract syntax, this will require reconsidering the semantics (step 2), which may be fed back again into the abstract syntax specification (step 5). This cycle $\langle 2;5\rangle$ may be repeated until the abstract syntax and its semantics are satisfactory and stable, at which point the annotation language is assumed to meet the requirement of semantic adequacy. The concrete syntax should now be adapted to this abstract syntax (step 3) - which in turn may have consequences that should be fed back (step 4). In fact, the 'outer cycle' $\langle 3;4\rangle$ does not make much sense to perform if not combined with the 'inner cycle' $\langle 2;5\rangle$, resulting together in the feedback loop (7):

(7) $\langle 4;\langle 2;5\rangle^*;3\rangle^*$

This feedback loop is particularly important not only for systematically developing a consistent design that starts from the initial stage of conceptual analysis, but also for being applied to a pre-existing representation format, in order to detect semantic deficiencies, or in order to develop an annotation language that meets the requirements of the ISO Linguistic Annotation Framework and the requirement of semantic adequacy; the latter applications will be discussed in Section 4.

3.1 From metamodel to abstract syntax

The definition of an abstract syntax consists of two parts: (1) the conceptual inventory, which specifies the basic concepts that may be used in annotations, and (2) the specification of the well-formed combinations of these elements as set-theoretical structures, the annotation structures. An annotation structure, as mentioned in Section 2.2, is of a set of (one or more) *entity structures* and (zero or more) *link structures*, as defined in (1) and (2), respectively.

Given a metamodel, the definition of an abstract syntax may follow the procedure outlined in (8):

(8) *Definition of abstract syntax from a given metamodel* (CASCADES step 1):

1. **Conceptual inventory:**
   Among the categories of concepts represented in the metamodel, identify those concepts that are basic and those that are composite, i.e., that are defined as combinations of other concepts occurring in the metamodel; the listing of the basic concepts constitutes the conceptual inventory.
2. **Annotation structures:**
   - For each category of composite concepts in the metamodel define an entity structure as a pair $\langle m, s\rangle$ consisting of (a) a markable m, and (b) the elements that characterise the instance s of the concept that is linked to that markable;
   - For each relational category define a link structure as a triple $\langle \varepsilon, E, \rho\rangle$ consisting of (a) the entity structure ε whose semantic relation to other entity structures is annotated; (b) the set E of these other entity structures; and (c) the semantic relation ρ (which may be a structured object, e.g. a pair $\langle m, r\rangle$ consisting of a markable m and a relation r).

Applied to the metamodel of Fig. 1 (which was inspired by the concept definitions of the DIT^{++} annotation schema (Bunt, 2009b) and indirectly by the DAMSL scheme (Allen & Core, 1997)), for defining a conceptual inventory we may observe that seven of the eight boxes in the metamodel correspond to categories of concepts to be used in annotations; the exception is the'dialogue' box, which represents a meta-concept, which is the source of the functional segments and contains the relevant metadata (not shown in Fig. 1). The concepts in six of these seven categories are basic; the only composite concept is the dialogue act. Of the six relations that occur in the metamodel, rhetorical relations is the only one which may have different values (like *Cause, Motivate*, or *Exemplify*) and therefore corresponds to a category of concepts; the labelled arrows represent single basic concepts.

The conceptual inventory of an abstract syntax lists the basic concepts in the metamodel; for DiAML the seven categories corresponding to boxes in Fig. 1 except the dialogue act box constitute categories of basic concepts; a dialogue act is a *composite concept*, composed of a sender, (an) addressee(s), possibly other participants, a content dimension, a communicative function, and possibly one or more qualifiers. A composite concept does not correspond to an item in the conceptual inventory, but to a structure built with objects from the conceptual inventory. The DiAML conceptual inventory thus lists six sets of basic concepts: (1) functional segments, (2) dialogue participants, (3) content dimensions, (4) communicative functions, (5) qualifiers, and (6) rhetorical relations, plus the relational concepts represented by labeled arrows in Fig. 1.

An entity structure in DiAML is one of the following nested pairs:

(9) a. $\varepsilon = \langle m, \langle \alpha, \Delta\rangle\rangle$

b. $\varepsilon = \langle m, \langle r, \langle\rangle\rangle\rangle$

made up of a markable m that identifies a functional segment (case 9a) or a text segment that expresses a rhetorical relation (case 9b); a 'dialogue act structure' α, which characterises a single dialogue act without the relations that it might have to other dialogue units; a 'dependence structure' Δ, which describes the semantic dependence relations between the dialogue act α and other dialogue units; and a rhetorical relation r.

A 'dialogue act structure' is a sixtuple

(10) $\alpha = \langle S, A, H, d, f, q\rangle$

where S is the sender of the dialogue act; A is a non-empty set of addressees; H is a (possibly empty) set of other dialogue participants (such as overhearers or side-participants; see Clark (1996); d is a dimension; f is a communicative function; and q is a (possibly empty) list of qualifiers. In order to avoid details which are irrelevant for this paper, we will only consider cases where the set H of participants who are neither speakers nor addressees is empty, and where there is only a single addressee - we will use A to indicate this addressee (rather than the set consisting of this addressee).

A 'dependence structure' is a pair consisting of a finite (possibly empty) set E of entity structures, whose members α has a dependence relation with, and R_{dep} specifying the dependence relation (functional or feedback - see below, Section 4.2):

(11) $\Delta = \langle E, R_{dep} \rangle$; $E = \{\varepsilon_1, \varepsilon_2, ..., \varepsilon_n\}$, $n \geq 0$; $R_{dep} = R_{func}$ or R_{fbck}

Link structures in DiAML are used to capture rhetorical relations between dialogue acts, for which qualifying information is not anticipated, hence the component q that link structures in general have (see (2)) is empty. Such relations may be expressed by discourse connectives but may also be implicit, as the metamodel in Fig. 1 indicates by a labelled arrow from rhetorical relations to functional segments with the specification '0..N' at the head. A link structure in DiAML is therefore one of the following two triples:

(12) a. $L = \langle \varepsilon, E, \langle r, \langle\rangle \rangle$
b. $L = \langle \varepsilon, E, \langle m, \langle r, \langle\rangle \rangle \rangle$

consisting of an entity structure ε, a finite set E of one or more entity structures, and a rhetorical relation r, possible expressed in the text by a markable m, which relates the dialogue act in α to those in E. (Note that in case 12b the link structure contains a relational entity structure of the type 9b.)

Since DiAML has just one category of composite concept, the dialogue act, and one type of relational concept, the rhetorical relation, the abstract syntax constructed in this way defines just one category of entity structure, which associates a dialogue act with a functional segment, and one type of link structure, which relates dialogue acts by a rhetorical relation.

It may be noted that the entity structures and link structures listed in (9) + (10 + (11) and (12) are not the only structures of the abstract syntax; additionally, several auxiliary structures are used, namely:

(13)
- dialogue act structures (see (10));
- dependence structures (see (11));
- sets of entity structures (see (11) and (12));
- relation structures (in (12b)).

In (15a) we see an example of an annotation structure, which applies to the dialogue fragment in (14), segmented into functional segments as shown in the lower part. Since there are no rhetorical relations in this fragment, the annotation structure consists just of three entity structures, one for each of the three dialogue acts expressed in the functional segments fs1, fs2 and fs3 (which all happen not to be multifunctional, and hence correspond to a single dialogue act). The column at the right in the lower part of (14) indicates the content dimensions in which the functional segments are relevant.

(14)

C: *Do you know what time the next train to Utrecht leaves?*	
S: *The next train to Utrecht leaves I think at 8:32.*	
fs1 = Do you know what time the next train to Utrecht leaves?	Task
fs2 = The next train to Utrecht	Auto-Feedback
fs3 = The next train to Utrecht leaves I think at 8:32.	Task

(15) $AS = \{\varepsilon_1, \varepsilon_2, \varepsilon_3\}$, with
$\varepsilon_1 = \langle fs1, \langle P_1, \{P_2\}, \phi, d1, F_4, \phi \rangle, \phi, - \rangle$
$\varepsilon_2 = \langle fs2, \langle P_2, \{P_1\}, \phi, d2, F_{29}, \phi \rangle, \{fs1\}, \text{feedback} \rangle$
$\varepsilon_3 = \langle fs3, \langle P_2, \{P_1\}, \phi, d1, F_7, \{c_2\} \rangle, \{\varepsilon_1\}, \text{functional} \rangle$

The entity structure ε_2 characterises the feedback act expressed in segment fs2, including the functional segment s_1 (of the entity structure ε_1) that it provides feedback about. The entity structure ε_3, which characterises the answer expressed in segment fs3, has the communicative function component F_7 ('Answer') and a qualifier component $\{c_2\}$ ('uncertain'); moreover, it includes the identifier ε_1 of the entity structure describing the question that the answer is functionally dependent on.

3.2 From abstract to concrete syntax

An ideal representation format in XML can be constructed systematically from a given abstract syntax using the procedure outlined in (16), which exploits the fact that entity structures are n-tuples of elements taken from the conceptual inventory, and link structures are made up of such entity structures plus an element from a relational category.

(16) *Definition of concrete syntax given an abstract syntax* (CASCADES step 3):

1. For each element of the conceptual vocabulary define an XML name;
2. For each type of entity structure $\langle m, s \rangle$ define an XML element with the following attributes and values:
   - the special attribute `@xml:id`, whose value is an identifier of the entity structure representation;
   - the special attribute `@target`, whose value represents the markable m;
   - attributes whose values represent the components of s_i, depending on the structure of s; if s_i is a basic concept then it is represented by its name.
3. For each type of link structure $\langle \varepsilon, E, \rho \rangle$ define an XML element with three attributes, two which have values that refer to the representations of the entity structures that are rhetorically linked, the value of the third denoting the relation between them.
4. For each type of auxiliary structure specify an XML representation. See the case of DiAML considered below.

In the case of DiAML, of which the definition of an abstract syntax was considered above, the auxiliary structures listed in (13) are used, which can be represented following the concrete syntax rules in (17):

(17)

1. A dialogue act structure $\alpha = \langle S, A, H, d, f, q \rangle$ is represented by a `dialogueAct` element which has six attributes whose values represent the six components of the dialogue act structure;
2. A dependence structure $\Delta = \langle E, R_{dep} \rangle$ occurring in an entity structure $\varepsilon = \langle m, \langle \alpha, \langle E, R_{dep} \rangle \rangle \rangle$ is not represented at all if E is empty; if E is not empty, then the dependence structure is represented by a sequence of two attributes in the representation of the dialogue act structure α, one having as its value the XML constant naming the dependence relation and the other representing the set E. If E is a singleton $\{\varepsilon_1\}$ then the latter value is just the representation of ε_1. (See the next item for the case that $E = \{\varepsilon_1, \varepsilon_2, ..., \varepsilon_n\}$ with $n \geq 2$.)
3. A set of entities $\{\varepsilon_1, \varepsilon_2, ..., \varepsilon_n\}$ with $n \geq 2$ is represented by an XML element (e.g. '`daGroup`'), which has an attribute whose values identify the members of E.
4. A relation structure of the form $\langle R, \langle\rangle \rangle$ occurring in a link structure $L = \langle \varepsilon, E, \langle R, \langle\rangle \rangle$, is represented by assigning the XML name of the relation R as value to the attribute in the link structure representation for representing the relation between the entity structure ε and those in E. For representational convenience, the XML element for representing link structures is given an additional optional attribute `@target`, so that this element can be used also for the case that the relation structure has the form $\langle m, R, \langle\rangle \rangle \rangle$.

Applying the procedure outlined in (16) together with (17) to the abstract syntax of DiAML leads to an ideal concrete XML-based syntax which defines the representation (18) for the example (14) - (15).

(18)
```
<diaml xmlns:"http://www.iso.org/diaml/">
 <dialogueAct xml:id="da1" target="#fs1"
   sender="#p1" addressee="#p2" dimension="task"
   communicativeFunction="setQuestion"
   conditionality="conditional"/>
 <dialogueAct xml:id="da2" target="#fs2"
   sender="#p2" addressee="#p1"
   dimension="autoFeedback"
   communicativeFunction="inform"
   feedbackDependence="fs1"/>
 <dialogueAct xml:id="da3" target="#fs3"
   sender="#p2" addressee="#p1" dimension="task"
   communicativeFunction="answer"
   certainty="uncertain" functionalDependence="#da1"/>
</diaml>
```

Note that each `dialogueAct` element in this representation corresponds to an entity structure in the abstract annotation structure; the value of the `@target` attribute identifies the functional segment that expresses the dialogue act; the value of the `@xml:id` attribute provides an identifier for the dialogue act structure as a whole, and the other attributes and values represent the basic concepts that the dialogue acts are composed of.

3.3 From abstract syntax to semantics

The construction of a formal semantics for a given abstract syntax is in general not straightforward, and depends on the form of semantics that is aimed for. If the annotations supported by the abstract syntax concern descriptive, propositional information, as is for instance the case with information about time and events, about semantic roles, about coreference, about spatial information, and about discourse relations, then a convenient form of semantics can be based on Discourse Representation Theory (DRT), as argued in Bunt (2012a). For the annotation of dialogue act information this is different, since an action-based semantics is more appropriate in this case (see Section 4.2.2).

For constructing a DRT-based semantics, given an abstract syntax, the CASCADES method provides the procedure outlined here, which is based on work reported in Bunt (2009a; 2011a; 2012a).

(19) *Definition of a DRT-based semantics for a given abstract syntax* (CASCADES step 2):

1. An entity structure $\langle m,s\rangle$ is interpreted as a DRS which:
   - introduces a discourse marker paired with a name of the markable m;
   - contains for each component s_i of s a condition $p_i(x,a_i)$, where a_i is the interpretation of the component s_i, p_i is a predicate that indicates the role of a_i, and x is the newly introduced discourse marker (examples below);
2. A link structure $\langle\langle m_1,s_1\rangle,E,\rho\rangle$ is interpreted as a DRS which:
   - introduces a discourse marker x_1, paired with a name of the markable m_1;
   - if E is a singleton $\{\varepsilon_2\}$, with $\varepsilon_2 = \langle m_2,s_2\rangle$, introduces a discourse marker x_2, paired with a name of the markable m_2;
     else if $E = \{\langle m_2,s_2\rangle,\langle m_3,s_3\rangle,...,\langle m_k,s_k\rangle\}$, introduces a discourse marker X_1, paired with a markable M_1 that identifies the (possibly discontinuous) stretch of primary data which is the merge of the stretches identified by the markables $m_2,m_3,...,m_k$;
   - introduces a condition of the form $R'_j(x_1,z)$, where R'_j is the interpretation of the relation in component ρ of the link structure; $z = x_2$ if $E = \{\varepsilon_2\}$; else $z = X_1$.
3. For each type of auxiliary structure that may occur within the s component of an entity structure $\langle m,s\rangle$ or within the ρ component of a link structure the DRS interpretation is specified.

Note the introduction of discourse markers paired with markables; this is to preserve the information that is contained in entity structures about the markable that certain semantic information is associated with. Once the interpretations of the individual entity structures and link structures, as represented by DRSs, have been combined to interpret the annotation structure as a whole, these markables are no longer needed, and the resulting DRS reduces to a classical one with ordinary, single discourse markers.

Illustrating this step in defining semantic annotations, Bunt (2012a) describes a semantics constructed for ISO-TimeML. For example, for annotating the temporal information in the sentence (20) the ISO-TimeML abstract syntax introduces an entity structure ε_1 for the *call* event and an entity structure ε_2 for the time point *midnight*, while the temporal anchoring relation between the event and the time point gives rise to a link structure L_1. The entity structure for an event is an n-tuple, where $1 \leq n \leq 6$, depending on the types of information available about the event. In this example only an event type and a tense are known, so the event structure is a pair $\langle$*event type, tense*$\rangle$.

(20) *John called Mary at midnight*

The semantics maps the entity and link structures to mini-DRSs as follows:

(21) $\varepsilon_1 \rightsquigarrow \begin{array}{|l|}\hline \langle m_2, e_1\rangle \\ \hline \textsc{evtype}(e_1, call) \\ \textsc{evperiod}(e_1, past) \\ \hline \end{array}$

$\varepsilon_2 \rightsquigarrow \begin{array}{|l|}\hline \langle m_4, t_1\rangle \\ \hline \textsc{clocktime}(t_1, 2400) \\ \hline \end{array}$

$L_1 \rightsquigarrow \begin{array}{|l|}\hline \langle m_2, e_1\rangle,\ \langle m_3, t_2\rangle \\ \hline \textsc{event-time}(e_1, t_2) \\ \hline \end{array}$

Merging these interpretations of the entity and link structures results in the following interpretation of the ISO-TimeML annotation:

(22) $\begin{array}{|l|}\hline \langle m_2, e_1\rangle, \langle m_4, t_1\rangle \\ \hline \textsc{evtype}(e_1, call) \\ \textsc{evperiod}(e_1, past) \\ \textsc{clocktime}(t_1, 2400) \\ \textsc{event-time}(e_1, t_1) \\ \hline \end{array}$

This says that a token of the *call* event type e_1 occurred in the past, at the time point t_1, which was at 24:00 hours.

An attractive feature of this kind of annotation semantics is that it can be applied equally to annotation structures capturing different kinds of information, allowing easy combination. For example, the annotation of semantic roles can be interpreted in the same way and integrated with the temporal information. In the case of example (21), the abstract syntax of the annotation of the semantic roles of *John* and *Mary* will introduce three entity structures, one (ε_3) for John, one (ε_5) for Mary, and one (ε_4) for the *call* event, and two link structures, one (L_2) for John playing the Agent role in the call event and one (L_3) for Mary playing the Theme role. The semantics of these structures takes the form of the following DRSs:

(23) $\varepsilon_3 \rightsquigarrow \begin{array}{|l|}\hline \langle m_1, x_1\rangle \\ \hline \textsc{person}(x_1, john) \\ \hline \end{array}$

$$\varepsilon_4 \leadsto \begin{array}{|l|}\hline \langle m_2, e_2\rangle \\ \hline \text{EVTYPE}(e_2, \mathit{call}) \\ \text{EVPERIOD}(e_2, \mathit{past}) \\ \hline \end{array}$$

$$\varepsilon_4 \leadsto \begin{array}{|l|}\hline \langle m_3, x_2\rangle \\ \hline \text{PERSON}(x_2, \mathit{mary}) \\ \hline \end{array}$$

$$L_2 \leadsto \begin{array}{|l|}\hline \langle m_1, x_1\rangle,\ \langle m_2, e_1\rangle \\ \hline \text{AGENT}(x_1, e_1) \\ \hline \end{array}$$

$$L_3 \leadsto \begin{array}{|l|}\hline \langle m_3, x_2\rangle,\ \langle m_2, e_1\rangle \\ \hline \text{THEME}(x_2, e_1) \\ \hline \end{array}$$

Merging the interpretations of the three entity structures and the two link structures gives us the interpretation of the semantic role information:

$$(24)\quad \begin{array}{|l|}\hline \langle m_1, x_1\rangle,\ \langle m_2, e_2\rangle,\ \langle m_3, x_2\rangle \\ \hline \text{PERSON}(x_1, \mathit{john}) \\ \text{EVTYPE}(e_2, \mathit{call}) \\ \text{EVPERIOD}(e_2, \mathit{past}) \\ \text{AGENT}(x_1, e_2) \\ \text{PERSON}(x_2, \mathit{mary}) \\ \text{THEME}(x_2, e_2) \\ \hline \end{array}$$

This interpretation indeed combines smoothly with that of the temporal information as represented in (22). Taking into account that the markables paired with discourse markers can be eliminated once the interpretations of all entity structures and link structures have been merged, we obtain through the merge of (22) and (24) the interpretation of both the temporal and semantic roles annotations:

$$(25)\quad \begin{array}{|l|}\hline x_1, e_1, x_2, t_1 \\ \hline \text{PERSON}(x_1, \mathit{john}) \\ \text{EVTYPE}(e_1, \mathit{call}) \\ \text{EVPERIOD}(e_1, \mathit{past}) \\ \text{AGENT}(x_1, e_1) \\ \text{PERSON}(x_2, \mathit{mary}) \\ \text{THEME}(x_2, e_1) \\ \text{CLOCKTIME}(t_1, 2400) \\ \text{EVENT-TIME}(e_1, t_1) \\ \hline \end{array}$$

The possibility to combine the semantics of different annotations is helpful for ensuring that annotation schemes concerned with different kinds of information are semantically compatible and ‘interoperable’.

4 Applications

In describing the CASCADES method in the previous section, in particular how the method helps to move from one design stage to another, we have in passing illustrated the use of the method for designing semantic annotations starting with a conceptual analysis. In practice, the design of a language for semantic annotation more often starts from an existing representational format or annotation practice. If that is to be the basis for developing an ISO standard, then an abstract syntax with a semantics must be constructed that fits the representations, in view of the requirements of the Linguistic Annotation Framework and the requirement of semantic adequacy, mentioned in Section 1. Starting from an existing practice, the CASCADES method can be used by following the double cycle $\langle 4; \langle 2; 5\rangle^{*}; 3\rangle$ (Fig. 3), starting with the reconstruction of an abstract syntax from the concrete syntax in step 4. (See also Section 3 where this feedback loop was introduced in (7)). The CASCADES method has been used in this 'reverse engineering' mode in the definition of an abstract syntax for the annotations in the Penn Discourse Treebank (PDTB) as a first step in the development of an ISO standard for the annotation of discourse relations (see Bunt, Prasad & Joshi, 2012).

Ide et al. (2011) have 'reverse-engineered' an abstract syntax for the PDTB representation format with the aim of designing a GrAF representation (Ide & Suderman, 2001) for these annotations, and have shown that, even without specifying a semantics for this abstract syntax, as would happen in the CASCADES feedback cycle $\langle 4; \langle 2; 5\rangle^{*}; 3\rangle$, this leads to improvements in the PDTB representations. They remark that *"The exercise of creating an abstract syntax for the PDTB scheme and rendering it in a graphic form shows the structure of the annotations more clearly. The concrete syntax is much more readable than the original format, and therefore errors and inconsistencies may be more readily identified."* Similarly, in designing a GrAF-based representation for the annotation of semantic roles in PropBank (Palmer et al., 2005), it was noted note that the existing annotation scheme is ambiguous as to the relations among the parts of an annotation; Ide & Bunt (2010), applying the combined ideas of abstract syntax and GrAF to a variety of existing annotation schemes, observe that *"The original PropBank encoding is close to an ideal concrete syntax, as it can be generated from the abstract syntax. However, the round trip back to the abstract syntax is not possible, because it is necessary to do some interpretation of associations among bits of annotations in order to construct the abstract syntax"*, and Ide & Suderman (2007) conclude that this is a demonstration of an *"all-too-pervasive feature of many annotation schemes: reliance on human interpretation"*.

Existing annotation schemes typically do not come with an annotation language for which a formal semantics is defined, and the distinctions made in the representations are therefore not always semantically well-founded, or the representations fail to make certain distinctions which should be made from a semantic point of view. Using the CASCADES method in reverse-engineering mode, in particular applying the double cycle $\langle 4; \langle 2; 5\rangle^{*}; 3\rangle^{*}$, can help to detect representational problems by making the underlying semantic issues explicit. Section 4.2 will illustrate this for two cases in the design of ISO standards for semantic annotation.

4.1 From the PDTB to ISO 24617-8

The annotation schema underlying the (PDTB, see Prasad et al., 2008) supports the annotation of the following aspects of semantic relations in discourse:

1. the discourse relations that define a semantic connection between two stretches of text (only binary relations are assumed);

2. the two stretches of text that describe the arguments of a discourse relation;

3. more extended stretches of text that may be useful for annotators for identifying or interpreting the arguments of a discourse relation, so-called 'supplemental arguments';

4. for two adjacent sentences which are not related by a discourse relation, if they form a coherent text fragment due to coreference relations then they are annotated as having the `EntityRel` relation; else they are annotated as having the relation `NoRel`;

5. for a discourse relation which is explicitly expressed in the text, whether the expression uses a discourse connective or is some other kind of expression;

    - if it is realised by an expression that uses a discourse connective, then:

    – which discourse connective is the lexical head of the expression;
    – which sense of the discourse connective is used
  - if the discourse relation is not expressed in the text, then which English discourse connective could be inserted to express it, and in which of its senses;

6. for discourse relations and their arguments the source that they are attributed to (such as the author of the text), with some further properties of the attribution (see below).

Some of these elements are illustrated in the following example, which shows the PDTB annotation of an implicit discourse relation that is not expressed in the text but which could have been realised by the connective *besides*. The relation is attributed to the source `"ot"` (which stands for *other*, as opposed to the author of the text or some arbitrary agent indicated in the text through a non-specific reference) and has the type `"comm"` (for 'communication', i.e. explicitly asserted, rather than e.g. an expressed belief); this attribution is expressed in the text segment with span `"726...752"` in the primary data (raw text files of the Wall Street Journal), located in the Penn Treebank at the Gorn address (Gorn, 1965) that refers to node sets in the treebank trees. The discourse relation is the comparative sense of the connective *besides*. Senses of connectives in the PDTB Form a 3-tier hierarchy, of which the top layer contains the four classes `Contingency, Temporal, Expansion,` and `Comparison`; within each of these classes various subclasses are defined, and for most subclasses certain elements are defined. For example, `Expectation` is an element of the `Concession` subclass of the `Comparison` class. The two arguments of the discourse relation are identified by their text spans and treebank addresses, and in this example are attributed to an 'inherited' source (`src="Inh"`).

(26)
```
<implicitRelation
  src="ot" type="Comm">
  <attr span="726...752"
    gorn="4, 0; 4, 1, 0; 4, 1, 1; 4, 1, 2; 4, 2">
  <ic sClass1="Expansion.Conjunction"
    besides</ic>
  <arg1 span="778...874" gorn="5" src="Inh"/>
  <arg2 span="876...916" gorn="6" src="Inh"/>
</implicitRelation>
```

As a first step in the direction of defining an ISO standard for discourse relation annotation, the cycle $\langle 4; \langle 2; 5\rangle^*; 3\rangle$ can be applied to construct an abstract syntax with a semantics, and an ideal concrete syntax for that abstract syntax. For performing the first step in this cycle, step 4, a procedure is required which does roughly the converse of step 3, so is it possible to invert the procedure (16) outlined for that step?

It may be observed first that some of the things that are marked up in the PDTB do not clearly constitute *semantic* information, such as whether a discourse relation is realised by a discourse connective or by a different kind of expression. Not being semantic in nature, this distinction cannot correspond to a distinction in the underlying abstract syntax, nor to a distinction in the metamodel. The distinction may be made in the concrete representations for other reasons, such as being helpful for annotators, or being of interest for other purposes than semantic interpretation, but for the reconstruction of an abstract syntax they can be left out of consideration. The following observations of this kind can be made about the PDTB annotations:

(27) 1. The senses in the hierarchy of discourse connective senses form in fact the set of discourse relations that are distinguished. Elements at the highest level of the hierarchy are coarse-grained discourse relations; those at the level below are finer-grained discourse relations; and those at the lowest level are the most fine-grained ones. When a discourse relation is annotated which is not one of the top four coarse-grained relations, then there is no need to explicitly annotate the higher levels, such as `Expansion.Conjunction` in (26); just `Conjunction` is sufficient.

2. The PDTB relations `NoRel` and `EntityRel` are not truly semantic discourse relations;
3. It is redundant to annotate whether a discourse relation is explicitly realised in the source text (this follows from whether the discourse relation is associated with a markable), and if it is, whether the realisation uses a discourse connective or some other form, like an adverbial phrase (the latter case represented in PDTB annotations by the `altLex` tag);
4. If a discourse relation is not explicitly realised, then it is redundant to specify a discourse connective that could be inserted; the specification of the sense of a connective that could be inserted is sufficient.
5. The 'supplemental arguments' of the PDTB annotation scheme are not clearly semantic in nature, and may from a strictly semantic point of view be considered to be redundant.

These observations can be taken into account[2] in two ways. One way is to change the XML representations, for example deleting inserted discourse connectives in the case of implicit discourse relations, in view of observation (27.4); another strategy, made possible by the distinction of an abstract and a concrete syntax, is to leave the XML representations as they are, but to refrain from introducing elements in the abstract syntax that would correspond to redundant or semantically irrelevant components. This latter possibility exploits the fact that an ideal representation format for a given abstract syntax, while requiring every annotation structure to be representable, and every representation to be unambiguous, does leave open the possibility that annotation structures have more than one representation. Applied to the annotation of implicit discourse relations, XML representations with and without an inserted connective can be considered as alternative representations of the same underlying abstract annotation structure. Similarly, the 'supplemental arguments' of the PDTB may be allowed in the XML representations, though not occurring in the underlying annotation structures.

CASCADES step 3 defines a maximally parsimonious concrete syntax for a given abstract syntax, which does not introduce any redundant or semantically irrelevant components, hence a 'reversal' of the procedure outlined in (16) for step 3 would not know what to do with such elements when they occur in a given concrete syntax. Applying the strategy of leaving a given concrete syntax untouched as far as redundant or irrelevant elements is concerned, but omitting corresponding elements in an underlying abstract syntax, we add a first step in the outline of a procedure for performing step 4 in (28) below, in which redundant and semantically irrelevant elements are identified in the concrete syntax, and provisions are made for how to omit them in the abstract syntax.

For defining the structures that an abstract syntax should generate, given a concrete syntax, it may be noted that the XML elements used by the concrete syntax can be divided into those that have a `@target` attribute with a markable as value, and those that do not have such an attribute. Entity structures being pairs $\langle m, s\rangle$, consisting of a markable m and some semantic information s (of which a particular abstract syntax defines the possible forms) correspond to the former kind of XML elements. Link structures, being of the general form $\langle \varepsilon, E, \rho\rangle$, do not refer directly to a markable, but may do so indirectly via their relational component ρ; link structures therefore correspond to the latter kind of XML elements. Both entity structures and link structures may contain auxiliary structures, as we have seen in some detail in Section 3.2 for the DiAML abstract syntax; these also correspond to XML elements of the latter kind, but they differ from the linking elements in that these have two attributes for identifying the two arguments of a relation, and a third attribute for identifying a relation between the arguments.

With these preliminaries, the following procedure for reconstructing an abstract syntax given a concrete syntax can be outlined:

(28) *Outline of procedure for defining an abstract syntax given a concrete syntax*:

1. Identify all those elements in the concrete representation format which are semantically redundant or irrelevant; depending on the nature of these elements (e.g. values of an XML

[2]The abstract syntax for the PDTB annotations developed here differs in many ways from the abstract syntax defined by Ide et al. (2011), because the aim in the latter case was to construct an abstract syntax which would account for everything in the PFDTB representations, including those elements which are semantically irrelevant. For example, their abstract syntax has different relations for textually explicit and implicit discourse relations.

attribute, or distinct XML element types) devise ways of treating the concrete syntax as if these elements or distinctions were not there.

2. For each type of linking element `L` defined by the concrete syntax define a link structure $\langle \varepsilon, E, \rho \rangle$ where ε is an entity structure, E is an auxiliary structure $\{\varepsilon_1, \ldots \varepsilon_n\}$ (a finite set of entity structures), and ρ is either a relational entity structure $\langle m, \langle r, q \rangle \rangle$ or an auxiliary structure $\langle r, q \rangle$. (See Section 2.2: the relational component of a link structure is either a pair $\langle r, q \rangle$ or a triple $\langle m, \langle r, q \rangle \rangle$, depending on whether the relation r is explicitly expressed in the primary data.)
3. For each type of non-linking XML element which has a `@target` attribute define an entity structure $\langle m, s \rangle$, such that:
   - m is a markable, corresponding to the value of `@target` in the XML-element;
   - s is an n-tuple $\langle s_1, \ldots s_k \rangle$, of which the components s_j correspond to the values of the attributes in the XML element, other than `@target` and `xml:id`.

Applied to representations in the PDTB, the first step in (28) is as follows:

(29)
1. The relations `NoRel` and `EntityRel` are left out of consideration.
2. Structured values of the `sClass` attribute, which represents a path in the sense hierarchy, are treated as consisting simply of the end point of the path.
3. The distinction between explicit and implicit discourse relations is disregarded.
4. Inserted discourse connectives in the annotation of implicit discourse relations are disregarded.
5. The annotations of lexical and syntactic properties of the textual realisation of an explicit discourse relation are disregarded.
6. The 'supplemental arguments' of PDTB annotations are disregarded.

Using the procedure outlined in (28) with the first step as in (29), we obtain an abstract syntax of which the conceptual inventory lists the semantically relevant basic concepts, and the annotation structures combine these ingredients as follows.

(30) a. Conceptual inventory:

- a finite set of markables, identifying the relevant segments in a given source text;
- a finite set of discourse relations;
- finite sets of elements corresponding to the values of the attribution attributes `@source`, `@type`, `@polarity`, and `@determinacy` (see Prasad et al. (2006) for details about these attributes);

b. Annotation structures:

- *Entity structures:* An entity structure in general being a pair $\langle m, s \rangle$, consisting of a markable m and semantic information s, the possible structures for s are:
  - *Relational entity structures:* $s = \langle r, a \rangle$ where r is a discourse relation and r is an attribution structure, which is an auxiliary structure in the form of a quadruple $\langle u, t, p, d \rangle$ consisting of a source, an attribution type, an attribution polarity, and an attribution determinacy;
  - *Argument structures:* s is an attribution structure, i.e. the entity structure is of the form $\langle m, a \rangle$, representing a markable which identifies a segment of source text that is the argument of a discourse relations, and its attribution.

- *Link structures:*
  A link structure is a triple $\langle \varepsilon_1, \{\varepsilon_2\}, \rho \rangle$ where ε_1 and ε_2 are argument structures and ρ is either a relational entity structure $\langle m, \langle r, a \rangle \rangle$ (used for annotating an explicit discourse relation) or an auxiliary structure $\langle r, a \rangle$ consisting of a discourse relation and an attribution structure (used for annotating an implicit discourse relation).

We can now apply the inner cycle $\langle 2;5 \rangle$ to check the semantic adequacy of this abstract syntax. If the abstract syntax is indeed adequate (see section 4.2 for applying the inner cycle), we apply step 3 and generate an ideal concrete syntax which is parsimonious and semantically adequate. According to the procedure for performing step 3 outlined in (16), this leads to the following concrete syntax, which additionally uses TEI-style markables as values of the `@target` attribute, rather than PDTB-style spans and Gorn addresses.

1. For representing argument structures an XML element `dRelArgument` is introduced.
2. For representing relational entity structures an XML element `dRel` is introduced.
3. For representing link structures an XML element `discourseRelation` is introduced, which replaces both the PDTB elements `implicitRel` and `explicitRel`.
4. For representing an attribution structure an XML element `attributionRep` is introduced.
5. For representing implicit relational structures $\langle r, a \rangle$, the `@target` attribute of the XML element type `dRel` is made optional, so that this element can be used for the representation of annotations of explicit as well as implicit discourse relations.

Applying this concrete syntax to the PDTB example (26), we obtain the representation (31), in which `m1` and `m2` are markables referring to the same locations in the source text as the PDTB values of the `@span` and `@gorn` attributes. Note that this representation does indicate that an *implicit* discourse relation is annotated, although the tag `discourseRelation` is used rather than the `implicitRelation` tag of the PDTB, by the fact that the relation represented by the `dRel` element does not refer to a markable. Should one wish to retain the possibility of the PDTB to insert a discourse connective that could have been used here, then that can be achieved by adding an additional attribute (say `'insertedConnective'`) in the `dRel` element. A value of this attribute would however be disregarded by the semantics; nothing corresponds to it in the abstract syntax.

(31)
```
<dRelML>
<discourseRelation xml:id="dr1"
  arg1="#a1" arg2="#a2" rel="#r1"/>
<dRelArgument xml:id="a1"
  target="#m1" attribution="#b1"/>
<dRelArgument xml:id="a2"
  target="#m2" attribution="#b21/>
<dRel xml:id="r1"
  relName="conjunction" attribution="#a1"/>
<attributionRep xml:id="b1"
  target="#m3" aSource="ot" aType="comm"/>
</dRelML>
```

4.2 Detecting and repairing deficiencies in annotation schemes

Existing annotation schemes have often been developed for a particular annotation task and are not so much a product of systematic design, but are driven by task- or domain-specific considerations and by characteristics of the primary data. As a result, some of the details of the annotation representations may not be satisfactory from a semantic point of view. The CASCADES method, in particular the double cycle (7) depicted in Fig. 3, can be used to bring such details to light, and indicate how they may be resolved

4.2.1 Recurring events in ISO-TimeML

ISO standard 24617-1 for annotating time and events (ISO, 24617-1:2012) includes a treatment of reference to recurring or quantified events, copied from their original treatment in TimeML (Pustejovsky, 2003), as illustrated by the annotation (33) of example (32):

(32) *John called twice*

(33)
```
<EVENT xml:id="e1" target="#token2" tense="PAST"/>
<TIMEX3 xml:id="t1 target="#token3 type="SET"
 freq="2X"/>
<TLINK eventID="#e1" relatedToTime="#t1"
 relType="DURING"/>
```

As pointed out by Bunt and Pustejovsky (2010) and Lee and Bunt (2012), this representation is not very satisfactory in several respects. First, 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*. But what kind of temporal entity is "`2X`" ? Is it a period, a date, a time, a measure, or a set? It does not fit any of these types, despite the indication `type="SET"`. In fact, *twice* should not be considered as denoting a temporal entity at all; it rather denotes a number, counting how many instances of a certain type of event occurred. Another problem is the use of the "`DURING`" relation between the event and the entity "`2X`". This relation holds between an event and a period, date or time during which it occurred, but the relation between an event and the number of its occurrences is not temporal in nature, hence "`DURING`" cannot be used here.

These problems are brought out very clearly when we apply the CASCADES cycle $\langle 4; \langle 2;5 \rangle^*; 3 \rangle$ to the concrete syntax of ISO-TimeML. Step 4 in the cycle immediately runs into a problem because the `<TIMEX3>` element does not correspond to a single entity structure in an abstract syntax, but to several different entity structures, depending on the value of the `@type` attribute. To take this special role of the `@type` attribute into account, the procedure for reconstructing an abstract syntax given a concrete syntax can be adapted accordingly. A more fundamental problem then appears: for cases like *twice* in (33) we get an entity structure of the form $\langle m, 2 \rangle$, which is clearly of a numerical rather than a temporal character. When the inner cycle $\langle 2;5 \rangle^*$ is applied to the reconstructed abstract syntax, we can hardly expect the entity structure $\langle m, 2 \rangle$ to be interpreted as anything else than the number 2, but that doesn't lead to a coherent semantics for the annotation structure underlying (33), since the link structure in (33) corresponds to a *temporal* relation between the entity structure for the event and the numerical entity $\langle m, 2 \rangle$.

From a conceptual point of view the sentence (32) describes a set of two tokens of the same type of event. Redesigning the abstract syntax so that it reflects this view, *twice* is interpreted as the cardinality of a set of events, corresponding in the abstract syntax to one of the elements in an entity structure describing such a set.

Based on the analysis of events (or, more generally, eventualities) in (Pustejovsky 2003; 2007), ISO 24617-1 characterises an event by nine features:

(34)
1. class (possible values: 'occurrence', 'reporting', 'perception', 'aspectual', 'state', 'intensional action', 'intensional state', copied from TimeML)
2. type (possible values: 'transition', 'state', 'process')
3. predicate (the lemma of a verb, noun or adjective)
4. part of speech
5. tense
6. aspect
7. verb form
8. modality
9. polarity

For example, the event description in (35) is annotated in ISO-TimeML as shown in (36):

(35) *John had not called by midnight*

```
      <EVENT xml:id="e1" target="#m2" ...  "#m4"
       pred="CALL"
(36)   class="OCCURRENCE" type="TRANSITION"
       pos="VERB" tense="PAST" aspect="NONE"
       vform="NONE" modality="NONE" polarity="NEG"/>
```

According to Bunt (2011a) this analysis can be simplified. First, the attributes `@pos` (part of speech) and `@vform` (verb form) are evidently not semantic in character. Second, the attribute `@pred` is semantically relevant only if its values are not just strings (as is the case in TimeML, where their value has the XML type 'CDATA'), but refer to elements in a repository of semantic concepts, e.g. an event ontology. Such a repository should contain such information as whether a *call* event is a state, a process, or a transition (the three possible values of `@type`), and whether it is a reporting event, a perception event, or an intentional state (three of the possible values of `@class`). Hence the specification of the 'class' and the 'type' of the event are redundant. An event therefore corresponds in the abstract syntax to an entity structure which is a 5-tuple, ⟨*predicate, tense, aspect, modality, polarity*⟩.

For dealing with sets of events, as in *John called twice*, ISO-TimeML has no provisions (only sets of temporal objects can be represented, using the `TIMEX3` element with the `type` attribute having the value `"SET"`). A minimal amendment to ISO-TimeML to make this possible is to introduce an attribute in `EVENT` elements that can have the value `"SET"`, and give this value a similar special role as the `@type` attribute in the `TIMEX3` element. We will call this attribute `@signature`. Moreover, when representing a set of events in this way we also want to be able to represent information about its cardinality, so we introduce an additional attribute `@cardinality`. The procedure for constructing an abstract syntax, adapted for the special role of `@signature`, introduces for `EVENT` elements with `@signature` value `"SET"` an entity structure which is a 7-tuple ⟨*predicate, tense, aspect, modality, polarity, signature, cardinality*⟩.

As before, the inner design cycle $\langle 2;5\rangle^*$ of the CASCADES model can be applied to check the semantic adequacy of the resulting abstract syntax. If the result of this check is positive (see section 4.2), application of the last step of the outer cycle $\langle 4;3\rangle$ generates an ideal concrete syntax which is able to represent repetitions of events in a semantically adequate way. For the example of *John called twice*, the result would be (omitting attributes with default values and with value `"NONE"`):

```
      <EVENT xml:id="e1" target="#token2"
(37)   type="CALL" tense="PAST"
       signature="SET" cardinality="2"/>
```

This representation illustrates the power of using ideal representation formats, saying in an optimally transparent way that two *call*-events occurred in the past. It is simpler than (33) and semantically more accurate.

4.2.2 Annotating functional and feedback relations in dialogue

As already mentioned in Section 2.1 in connection with the metamodel for dialogue act annotation in Fig. 1, and briefly considered in Section 3.1, ISO standard 24617-2 for dialogue act annotation includes not only the annotation of dialogue acts, but also of relations between dialogue acts. Three types of relations are distinguished:

1. *functional dependence* relations, which relate dialogue acts that are responsive in nature, such as Answer, Confirmation, Agreement, Accept Apology, and Decline Offer to the dialogue act that they respond to;

2. *feedback dependence* relations, which relate a feedback act, i.e. a dialogue act which provides or elicits information about the processing of something that was said before, to the relevant elements in the dialogue history;

3. *rhetorical relations*, which indicate semantic relations like Explanation, Elaboration, or Cause between dialogue acts or their contents.

Responsive dialogue acts have a semantic content that depends crucially on the dialogue act they respond to, and it is perhaps not a coincidence that they can be expressed by utterances that by themselves have little or no semantic content, such as *"Yes", "No thanks", "No problem",* and *"OK"*. The marking up of functional dependence relations in DiAML offers the possibility to for example annotate a functional segment not only as expressing an answer, but also indicating which question it answers.

DiAML was developed systematically using the CASCADES approach, defining a metamodel in a first stage of conceptual analysis, and subsequently an abstract syntax. Note that, according to the steps and stages of CASCADES shown in Fig. 3, the definition of a semantics and a representation format for the abstract syntax in steps 2 and 3 can be done in any order. An advantage of first taking step 2 and even better the cycle $\langle 2;5\rangle^*$, is that the abstract syntax is checked for being semantically adequate before effort is spent on designing a representation for it. On the other hand, simultaneously performing steps 2 and 3 and subsequently applying the cycle $\langle 4;\langle 2;5\rangle^*;3\rangle$ may be quicker, since a first version of the representation format can be ready when the semantics is defined, and the cycle $\langle 4;\langle 2;5\rangle^*;3\rangle$ can be performed as a check of semantic adequacy, which will typically lead to relatively minor changes in the representation. In the case of DiAML the definition of a semantics and of a representation format were constructed in parallel. The preliminary version ISO DIS 24617-2:2010 of the standard (henceforth: 'DIS-DiAML'), corresponds to a stage where an abstract and a concrete syntax had been defined as well as the semantics of entity structures, but where the semantics of link structures was not fully defined.

Example (39) shows the representation of a functional dependence relation according DIS-DiAML, of the dialogue fragment (38)[3], using a `functionalLink` element to represent the dependence between question and answer (where the answer is expressed by the discontinuous functional segment *"No (..) there isn't"*).

(38)
1. C: *Is there an earlier connection?*
2. A: *No, I'm sorry, there isn't.*

(39)
```
<diaml xmlns:
  "http://www.iso.org/diaml/"/>
<dialogueAct xml:id="e1" target="#fs1"
  sender="#c" addressee="#a"
  communicativeFunction="propositionalQuestion"
  dimension="task"/>
<dialogueAct xml:id="e2" target="#fs2"
  sender="#a" addressee="#c"
  communicativeFunction="answer" dimension="task"/>
<functionalLink
  dact="#e2"
  functionalAntecedent="#e1"/>
<dialogueAct xml:id="e3" target="#fs3"
  sender="#a" addressee="#c"
  communicativeFunction="apology"
  dimension="social obligations"/>
</diaml>
```

In the abstract syntax of DIS-DiAML, all relations between dialogue acts would be annotated by means of link structures, so the annotation structure for (38) consists of three entity structures ε_1, ε_2,

[3]From the OVIS corpus, see `http://www.let.rug.nl/~vannoord/Ovis`.

and ε_3, corresponding to the question, the answer, and the apology, respectively, and one link structure L_1 for the functional relation between question and answer.

(40) $AS = \{\varepsilon_1, \varepsilon_2, \varepsilon_3, L_1\}$, with $L_1 = \langle \varepsilon_2, \{\varepsilon_1\}, \langle R_{fu}, \langle\rangle\rangle\rangle$

Application of CASCADES step 2 and developing a formal semantics which takes the various possible relations between dialogue acts fully into account, reveals that this is not a semantically adequate annotation.

Underlying the analysis of dialogue in terms of dialogue acts is the 'information-state update' approach, which views dialogue acts semantically as updates of the information states of dialogue participants (Bunt, 1989; 2000; 2011b; Poesio & Traum, 11997; 998; Traum & Larsson, 2003; Petukhova, 2011). On this approach, each type of dialogue act corresponds to a particular update operation. The semantics of an annotation structure $\{\varepsilon_1, .., \varepsilon_n, L_1, .., L_k\}$, consisting of the entity structures $\{\varepsilon_1, .., \varepsilon_n\}$ and the link structures $\{L_1, .., L_k\}$, is defined as the successive application of the update operations corresponding to each of the entity and link structures, ordered by the textual order $<_T$ of their functional segments, where the update operations corresponding to textually coinciding ('$=_T$') entity structures are unified rather than sequenced. The notation $;/\sqcup$ is used to indicate this: formally, '$\alpha\ ;/\sqcup\ \beta$' means that the operation α should be followed (';') by the operation β if $\alpha <_T \beta$; if $\alpha =_T \beta$ then the two operations should be unified ($\sqcup$). (See Bunt, 2011b; 2012b for details and examples.)

(41) $I_a(\{\varepsilon_1, .., \varepsilon_n, L_1, .., L_k\}) = I_a(e_1)\ ;/\sqcup\ ...;/\sqcup\ I_a(e_n)\ ;/\sqcup\ I_a(L_1)\ ;/\sqcup\ ...\ ;/\sqcup\ I_a(L_k)$

Successive application of the information state update operations for each of these structures runs into the problem, however, that the update operation for ε_2 cannot be defined independently of the link structure L_1, since the link to the answer's question is needed for determining the semantic content that is negated by A saying *"No it isn't"*.

Applying the CASCADES cycle $\langle 5;2\rangle$ to repair this, we may note that the source of the problem is that the semantic information about the answer is split up into a 'local' part in the entity structure ε_2 and the functional dependence part in the link structure L_1. These two parts should not be separated; the functional dependence information is an inseparable part of the semantic characerisation of an answer and should thus be in the entity structure. Rather than (40), the annotation structure for (38) should therefore be (42):

(42) $AS' = \{\varepsilon_1, \varepsilon_2, \varepsilon_3\}$ with $\varepsilon_2 = \langle m_2, \langle P_2, P_1, \emptyset, d_1, F_7, \langle\rangle\rangle, \langle\{\varepsilon_1, R_{fu}\rangle\rangle$

For feedback dependence relations the situation is essentially the same as for functional dependence relations: the semantics of a feedback act crucially depends on what previous element(s) in the dialogue the feedback is about; therefore a feedback dependence should be treated as being part of the entity structure that characterises the feedback act, rather than as a link structure (as in DIS-DiAML).

For rhetorical relations the situation is different, since they provide information about why a dialogue act occurs, rather than being part of the semantics of the dialogue acts. A link structure $L = \langle \varepsilon, E, \rho\rangle$ is interpreted semantically as a set of updates that create rhetorical links between the representations in the participants' information states of the dialogue acts in ε and E.

Applying CASCADES step 3 to the revised abstract syntax, we obtain a semantically adequate ideal representation format, illustrated by the representation (43) for example (38), in which the entity structure ε_2 contains the attribute `@functionalDependence` whose value specifies the question to which the dialogue act in ε_2 forms the answer.

(43)

```
<diaml xmlns:
  "http://www.iso.org/diaml/"/>
<dialogueAct xml:id="e1" target="#fs1" sender="#c"
  addressee="#a" dimension="task"
  communicativeFunction="propositionalQuestion"/>
<dialogueAct xml:id="e2" target="#fs2"
  sender="#a" addressee="#c"
  communicativeFunction="answer" dimension="task"
  functionalDependence="#e1"/>
<dialogueAct xml:id="e3" target="#fs3" sender="#a"
  addressee="#c" communicativeFunction="apology"
  dimension="social obligations"/>
</diaml>
```

5 Conclusions

The CASCADES approach to the design of semantic annotation languages brings three main innovations.

First, compared to the usual way of defining a formal language, a component has been added which specifies the categories of information that can be expressed in the language, and the ways in which elements of these categories may be combined. This 'abstract syntax' specification is in set-theoretical terms, independent of any representation format. What is traditionally called a syntax, by contrast, concerns a particular representation format, and corresponds with a 'concrete syntax' in this approach. This distinction is particularly important in the context of defining annotation standards, since according to the ISO Linguistic Annotation Framework, standards should be defined at an abstract level, independent of any representation format. The introduction of an abstract syntax layer in the definition of an annotation language supports a principled view on the information that annotations are intended to capture, and is helpful in the design of well-founded semantic annotation standards, starting from a conceptual analysis, and resulting in semantically interpreted, maximally simple and transparent representations.

Second, it has been shown that the semantics of an annotation language can be defined 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 same semantics from the abstract syntax; this is beneficial for improving the interoperability of semantically annotated corpora that use different representation formats.

Third, the notion of an 'ideal representation format' has been introduced for a format which is able to represent every annotation structure defined by the abstract syntax, and where each representation is the rendering of exactly one annotation structure. It was shown that any two ideal representation formats can be converted to each other through a strictly meaning-preserving mapping. Ideal representation formats have the advantage of being maximally *simple*, representing only the conceptual distinctions made in the abstract syntax and the underlying metamodel, and being optimally *transparent*, having a direct correspondence with conceptual distinctions.

In addition to these conceptual innovations, the CASCADES methodology comes with outlines of procedures for going systematically from stage of annotation design to another. A conceptual analysis of an annotation domain, expressed in the form of a metamodel, was shown to allow the systematic construction of an abstract syntax, and a given abstract syntax was shown to allow the systematic construction of an ideal XML-based concrete syntax, defining a particular representation format. Moreover, a given abstract syntax was shown to allow the systematic definition of a DRT-based semantics for those domains where this form of semantics is appropriate. The CASCADES methodology thus offers a systematic process for developing a semantic annotation language, starting with a conceptual analysis.

The procedures for stepping from design stage to another can more or less be inverted, which is especially interesting for constructing an abstract syntax from a concrete syntax rather than the other way round. This makes the design of semantic annotations using the CASCADES stages more realistic, both for allowing systematic feedback cycles and for allowing the process to start from an existing annotation format. This was illustrated for the annotation of discourse relations in text, starting from the Penn Discourse Treebank.

These feedback cycles make the CASCADES model also useful for detecting deficiencies in existing or developing annotation schemes and pointing the way for how to resolve them. This was illustrated for problems in the annotation of recurring events in ISO-TimeML and of dependence relations in dialogue using DiAML.

Acknowledgements

The establishment of the CASCADES methodology has benefited greatly from discussions with colleagues in the business of developing standards for interoperable annotations, and from application of the methodology in ISO projects that aim to define international standards for semantic annotation. I would like to thank in particular Nancy Ide, James Pustejovsky, Rashmi Prasad, Kiyong Lee, Laurent Romary, Volha Petukhova, David Traum, Jan Alexandersson, Andrei Popescu-Belis, Martha Palmer, Alex Fang, and Aravind Joshi.

I would also like to thank two anonymous reviewers, whose insightful and detailed comments have been very important for bringing this paper in its present form.

References

Allen J. and M. Core (1997) DAMSL: Dialogue Act Markup in Several Layers (Draft 2.1). Technical Report. University of Rochester, Rochester, NY.

Bonial, C., S. Windisch Brown, W. Corvey, M. Palmer, V. Petukhova and H. Bunt (2011b) An Exploratory Comparison of Thematic Roles in VerbNet and LIRICS. In *Proc. 6th Joint ACL-ISO Workshop on Interoperable Semantic Annotation ISA-6*, Oxford

Bonial, C., W. Corvey, M. Palmer, V. Petukhova and H. Bunt (2011b) A Hierarchical Unification of LIRICS and VerbNet Semantic Roles. In *Proc. ICSC Workshop on Semantic Annotation for Computational Linguistic Resources (SACL-ICSC 2011)*, Stanford.

Bunt, H. (1989) Information dialogues as communicative action in relation to partner modelling and information processing. In: M. Taylor, F. Néel, and D. Bouwhuis D (eds) *The structure of multimodal dialogue.* North-Holland, Amsterdam, pp. 47–74.

Bunt, H. (2000) Dialogue pragmatics and context specification. In: H. Bunt and W. Black (eds) *Abduction, Belief and Context in Dialogue.* Benjamins, Amsterdam, pp. 81–150.

Bunt, H. (2009a) Semantic Annotation as Complementary to Underspecified Semantic Representations. In: *Proc. IWCS 2009, the Eighth International Conference on Computational Semantics*, Tilburg, pp. 33–44.

Bunt H. (2009b) The DIT^{++} taxonomy for functional dialogue markup. In: *Proc. of EDAML/AAMAS Workshop "Towards a Standard Markup Language for Embodied Dialogue Acts*, Budapest, pp. 13–24. Available (with updates) at `http:///dit.uvt.nl`

Bunt, H. (2010) A Methodology for designing semantic annotation languages exploiting syntactic-semantic iso-morphisms. In: A. Fang, N. Ide, and J. Webber (eds.) *Proc. ICGL 2010, the 2nd International Conference on Global Interoperability for Language Resources*, Hong Kong, pp. 29–45.

Bunt, H. (2011a) Introducing abstract syntax + semantics in semantic annotation, and its consequences for the annotation of time and events. In E. Lee and A. Yoon (eds), *Recent Trends in Language and Knowledge Processing*. Hankukmunhwasa, Seoul, pp. 157-204.

Bunt, H. (2011b) The Semantics of Dialogue Acts. In: *Proc. of IWCS 2011, the Ninth International Conference on Computational Semantics*, Oxford, pp. 1–14.

Bunt, H. (2012a) Annotations that effectively contribute to semantic interpretation. In: H. Bunt, J. Bos and S. Pulman (eds.) (forthc.) *Computing Meaning, Vol. 4.* Springer, Berlin, pp. 49-72.

Bunt, H. (2012b) A context-change semantics for dialogue acts. In: H. Bunt, J. Bos and S. Pulman (eds.) (forthc.) *Computing Meaning, Vol. 4*. Springer, Berlin, pp. 145-167.

Bunt, H. and J. Pustejovsky (2010) Annotation of temporal and event quantification. In: *Proc. Fifth Joint ACL-ISO Workshop on Interoperable Semantic Annotation ISA-5*, Hong Kong, pp. 15–22.

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

Bunt, H. and L. Romary (2004) Standardization im Multimodal Content Representation: Some Methodological Issues. In: *Proc.* 4th *International Conference on Language Resources and Evaluation (LREC 2004)*, Lisbon. ELRA, Paris, pp 2219–2222.

Bunt, H, J. Alexandersson, J. Carletta, J.-W. Choe, A.Fang, K. Hasida, K. Lee, V. Petukhova, A. Popescu-Belis, L. Romary, C. Soria, and D. Traum (2010) Towards an ISO standard for dialogue act annotation. In: *Proc.* 7th *International Conference on Language Resources and Evaluation (LREC 2010)*, Malta. ELRA, Paris.

Bunt, H, J. Alexandersson, J.-W. Choe, A.Fang, K. Hasida, K. Lee, V. Petukhova, A. Popescu-Belis, L. Romary, and D. Traum (2012) A semantically-based standard for dialogue annotation. In: *Proc. 8th International Conference on Language Resources and Evaluation (LREC 2012)*, Istanbul. ELRA, Paris.

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

Gorn, S. (1965) Explicit definitions and linguistic dominoes. In J. Hart and S. Takasu (eds) *Systems and Computer Science*. University of Toronto Press, Toronto, Canada.

Ide, N. and H. Bunt (2010) Anatomy of Annotation Schemes: Mappings to GrAF. In: *Proc. of LAW-IV: the Fourth Linguistic Annotation Workshop,* Uppsala, pp. 115–124.

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

Ide, N. and L. Romary (2006) Representing Linguistic Corpora and their Annotations. In: *Proc. 5th International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa. ELRA, Paris.

Ide, N. and K. Suderman (2007) GrAF: A Graph-based Format for Linguistic Annotations. In *Proc. First Linguistic Annotation Workshop*, Prague, pp. 1–8.

Ide, N., R. Prasad and A. Joshi (2011) Towards Interoperability for the Penn Discourse Treebank. In *Proc. 6th Joint ACL-ISO Workshop on Interoperable Semantic Annotation (ISA-6)*, Oxford, pp. 49-55.

ISO-DIS 24617-2:2010 *Language Resource Management - Semantic Annotation Framework, Part 2: Dialogue Acts*, Draft International Standard. ISO, Geneva.

ISO 24612:2012 Language Resource Management: Linguistic annotation framework, ISO, Geneva.

ISO 24617-1:2012 *Language Resource Management - Semantic AnnotationFramework, Part 1: Time and Events*, International Standard. ISO, Geneva, ISO 24617-2:2012 *Language Resource Management - Semantic Annotation Framework, Part 2: Dialogue Acts*, International Standard. ISO, Geneva.

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

Lee, K. and H. Bunt (2012) Counting time and events. In *Proc. of "ISA in Pisa", the 8th Joint ACL - ISO Workshop on Interoperable Semantic Annotation (ISA-8)*, Pisa, pp. 34-42.

Palmer, M., D. Gildea, and P. Kingsbury (2005) The Proposition Bank: An Annotated Corpus of Semantic Roles, *Computational Linguistics* 31 (1), 71-106.

Petukhova, V. (2011) *Multidimensional Dialogue Modelling.* Ph.D. Thesis, Tilburg University.

Petukhova, V. and H. Bunt (2010) Introducing communicative function qualifiers. In: A. Fang, N. Ide, and J. Webber (eds.) *Proc. of ICGL 2010, the Second International Conference on Global Interoperability for Language Resources*, Hong Kong, pp. 123–131.

Poesio M. and D. Traum (1997) Conversational actions and discourse situations. Computational Intelligence 13(3), 309 – 347.

Poesio M, Traum D (1998) Towards an axiomatization of dialogue acts. In: J. Hulstijn and A.Nijholt (eds) *Proc. Twente Workshop on the Formal Semantics and Pragmatics of Dialogues*, Enschede, pp. 207-222.

Prasad, R., N. Dinesh, R. A. Lee, A. Joshi and B. Webber (2008) Annotating Attribution in the Penn Discourse Treebank. In *Proc. ACL 2006 Workshop on Sentiment and Subjectivity in Text*, Sydney, pp. 31-38.

Prasad, R., N. Dinesh, A. Lee, E. Miltsakaki, L. Robaldo, A. Joshi and B. Webber (2008) The Penn Discourse Treebank 2.0. In *Proc. 6th International Conference on Language Resources and Evaluation (LREC 2008)*, Marrakech.

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

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

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

Pustejovsky, J., K. Lee, H. Bunt, and L. Romary (2010) ISO-TimeML: An International Standard for Semantic Annotation. In: *Proc. Sevenfth International Conference on Language Resources and Evaluation (LREC 2010)*, Malta. ELRA, Paris, pp. 394–397.

TEI (2009) *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. Lou Burnard and Syd Bauman, editors. Text Encoding Initiative Consortium, Oxford.

Traum D. and S. Larsson (2003) The information state approach to dialogue management. In: J. van Kuppevelt and R. Smith) (eds) *Current and New Directions in Discourse and Dialogue*, Kluwer, Dordrecht, pp. 325–345.