Generating Conceptually Aligned Texts

David Hardcastle
Faculty of Maths, Computing
And Statistics
The Open University
Milton Keynes, UK
d.w.hardcastle@open.ac.uk

Richard Power
Faculty of Maths, Computing
And Statistics
The Open University
Milton Keynes, UK
r.power@open.ac.uk

Abstract

A conceptually aligned text is one in which spans are systematically linked to a logical encoding of their meanings. We describe the model of alignment used in WYSIWYM systems, which support knowledge editing through an automatically generated feedback text, and show that this model has a series of practical advantages through which it can support applications requiring semantic interactivity and inference, as envisaged by the Semantic Web community. Some results from an efficient recent implementation are presented, showing the benefits of having all levels of linguistic description available for interactive services.

1 Introduction

We define a *conceptually aligned* text as one in which linguistic units are systematically linked to a formal encoding of their meanings. This definition raises theoretical issues that have troubled linguists and logicians since antiquity (Percival, 1990) and are far from resolved today. However, the explicit linking of text to meaning can also be seen as a *practical* issue: there might be alignment methods that are useful for supporting interactions between user and text, even though from a theoretical perspective they are (at best) oversimplified.

The main practical benefit of aligning text with meaning is that a user can manipulate formally-encoded knowledge *through* the text, so obtaining transparent access to knowledge-based services. One example, which we will discuss later, is the WYSIWYM method for editing formally encoded knowledge by interacting with a computer-generated 'feedback text' (Power and Scott, 1998; Power et al., 2003). More recently, the idea of exploiting links between text and semantic data has been promoted by the Semantic Web community (Berners-Lee et al., 2001), with applications including semantic search and querying.

Since automatic interpretation of text is unreliable, and manual annotation tedious, the most convenient means of producing conceptually aligned text is currently Natural Language Generation (NLG), for the obvious reason that the formally encoded meaning is already available to the system as input. It is curious how little the NLG community has noticed this benefit and exploited it by applications that allow semantic interactivity: to our knowledge, WYSIWYM knowledge editing remains the only family of NLG applications in which the links between linguistic units and their meanings are preserved and used.

Our aim in this paper is to describe and motivate the model of conceptual alignment implemented in WYSIWYM systems. This model depends on a data structure called the 'Atree', which mediates between meaning and text. We define the Atree and describe a Java implementation in which it links a syntactic representation in TAG (Tree Adjoining Grammar) with a semantic representation in Description Logic (section 2). We then review the advantages of the alignment model based on the Atree (section 3), and show how it can support applications requiring semantic interactivity (section 4). The paper concludes (section 5) by discussing the significance of

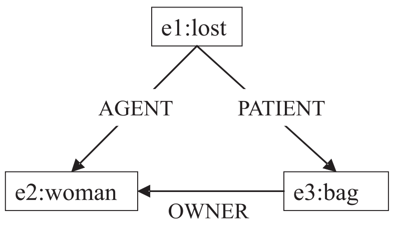


Figure 1: Sample Abox

these ideas for research on Controlled Languages and the Semantic Web.

2 Alignment method

2.1 Basic principles

WYSIWYM was originally developed as a way of defining the input for multilingual NLG in DRAFTER (Paris et al., 1995), one of a series of projects in the KPML/Penman tradition (Bateman et al., 1989). The semantic representation was a version of what would now be called Description Logic, with the usual distinction between Tbox (ontology) and Abox (factual assertions about domain entities). This assumption has remained a constant feature of WYSIWYM applications (Hallett et al., 2007), and has also become a standard through the Semantic Web. An Abox is a set of assertions defining relations between typed entities. It can be conveniently depicted by a connected graph (figure 1) in which vertices represent entities and edges represent relations; equivalently it can be written down in predicate-argument form, with one-place predications assigning types and two-place predications asserting relationships:

LOST(e_1) & AGENT(e_1,e_2) &
WOMAN(e_2) & PATIENT(e_1,e_3) &
BAG(e_3) & OWNER(e_3,e_2)

Assuming that the entities are being mentioned for the first time, we might express this Abox fragment in English by the sentence ‘a woman lost her bag’. This sentence can be aligned with the Abox by following two principles, one concerning entities, the other concerning relationships:

1. Entities are represented by spans of text. If an entity occurs in several roles, each occurrence will be represented by a separate span.

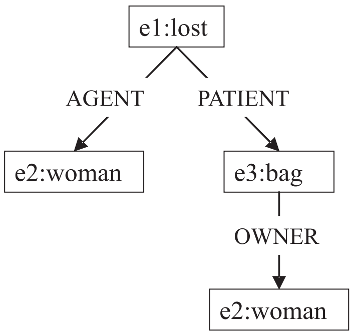


Figure 2: Sample Atree

2. Relationships between entities are represented by span-subspan relations in the text. If a relation R(e_1,e_2) holds between e_1 and e_2, then the span realising e_2 in the role R will be a constituent of the span realising e_1.

Applying these rules to the example, we can identify four spans denoting entities:

Span	**Entity**	**Role**
a woman lost her bag	e_1	ROOT
a woman	e_2	AGENT
her bag	e_3	PATIENT
her	e_2	OWNER

In the obvious way, all three relationships are represented by span-subspan relations that can be read off this table. For instance, AGENT(e_1,e_2) is realised by the clause-subject relation between ‘a woman lost her bag’ and its subspan ‘a woman’. This direct linking of semantic and syntactic dependencies has of course been noted many times, for instance in Meaning-Text Theory (Candito and Kahane, 1998).

Underlying any such span structure is a reconfiguration of the original Abox as an ordered tree, which we will henceforth call an *Atree*. Figure 2 shows an Atree that fits our example (and kindred realisations). Note that since this is a tree, the vertex with two incoming edges (e_2) has to be repeated, so that we get two spans referring to the woman. Edges now indicate syntactic dependencies, so that daughters represent subspans of their parent, and the left-to-right order of daughters is significant — a different Atree would be required for the (rather ugly) rendition ‘a woman’s bag was lost by her’.

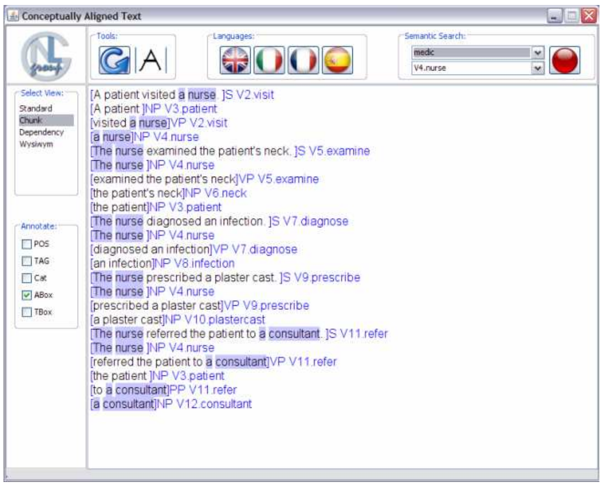
For WYSIWYM applications the Atree is significant because it maps spans of the generated text to entities (and roles) in the Abox; operations for editing the Abox can then be offered through menus opening on these spans (Power and Scott, 1998). However, the Atree also has a wider potential significance as an interesting stage of planning in NLG generally. Like the Roman god Janus it looks in two directions: one face looks to the meaning, because an Atree is a notational variant of an Abox; the other face looks to the text, because an Atree expresses basic realisational decisions concerning dependency and linear order.

2.2 Simplifying assumptions

We plan to elaborate elsewhere on the theoretical applications of the Atree (e.g., as a way of enumerating the dependency structures that can be derived from a given Abox). In the present paper, however, our concern is with practical applications such as WYSIYWM editing, which make special demands on speed and reliability. To meet these requirements, our current implementations make simplifying assumptions which minimise search in all stages of generation, thus promoting efficiency and scalability.

We assume, firstly, that for any complex document, the upper levels of the Abox will take the form of a tree that maps directly to a document structure of sections, paragraphs, etc. (Power et al., 2003). This property is enforced by the top-down editing procedure in WYSIWYM applications, starting with a label that embraces the whole text, and allowing coreference only when the semantic level descends to particular events, people and objects. The benefit from this assumption is that text structuring becomes trivial, since the organisation of the semantic material has already been encoded in the Abox through user decisions; the cost is that the Abox is littered with assertions that really concern textual organisation rather than knowledge about the domain (of course these could easily be removed before using the Abox for other purposes).

Our second simplifying assumption is that the text should adhere strictly to a controlled language, so that a given local semantic configuration is always realised by the same linguistic pattern. This yields obvious gains in efficiency and consistency, at the expense of a potentially clumsy and repetitive style. This assumption allows an efficient realisation algorithm in which a TAG derivation tree is computed by labelling each vertex in the Atree with a TAG elementary tree (more details follow).



```
[A patient visited a nurse. ]S V2.visit
[A patient ]NP V3.patient
[visited a nurse]VP V2.visit
[a nurse]NP V4.nurse
[The nurse examined the patient's neck. ]S V5.examine
[The nurse ]NP V4.nurse
[examined the patient's neck]VP V5.examine
[the patient's neck]NP V6.neck
[the patient]NP V3.patient
[The nurse diagnosed an infection. ]S V7.diagnose
[The nurse ]NP V4.nurse
[diagnosed an infection]VP V7.diagnose
[an infection]NP V8.infection
[The nurse prescribed a plaster cast. ]S V9.prescribe
[The nurse ]NP V4.nurse
[prescribed a plaster cast]VP V9.prescribe
[a plaster cast]NP V10.plastercast
[The nurse referred the patient to a consultant. ]S V11.refer
[The nurse ]NP V4.nurse
[referred the patient to a consultant]VP V11.refer
[the patient ]NP V3.patient
[to a consultant]PP V11.refer
[a consultant]NP V12.consultant
```

Figure 3: A screenshot of the demonstration application

2.3 Implementation

The Atree model of alignment has been implemented in a Java demonstrator application (shown in Figure 3) which generates WYSIWYM feedback texts in English, Spanish, French and Italian for Aboxes representing simplified narratives of patient-doctor interactions. The application supports semantic search — the user can highlight spans that relate to any given Tbox concept or Abox instance — as well as mouse-sensitive interaction with the text as described in previous WYSIWYM literature (Power and Scott, 1998). It can also provide views of the text with syntactic information (POS tags, parsing) or semantic instances, concepts or roles, thus showing that all levels of information have been retained in a linked data structure.

The Tbox is represented as a tree with pointers defining the attributes of each concept. The Abox is a directed graph with a single root node; each node in the graph is an instance of some concept defined in the Tbox, and each edge expresses an attribute of that concept, as discussed above. The relationship between Abox and Tbox is represented through pointers from the nodes and edges of the Abox back onto the concepts and attributes in the Tbox, so that the Abox is a connected extension of the Tbox data structure in memory. Given a Tbox, an Abox and a target natural language, the system further extends the data structure by adding first an Atree connected to the Abox, and then a derived tree based on the TAG formalism which is connected to the Atree.

The resulting data structure represents the text (which can be read off the derived tree by left-right traversal), information relating to various linguistic levels of the text (such as morphology, syntax, semantic roles and dependencies), the underlying semantics of the text (drawn from the instances and relationships specified in the Abox) and the conceptual framework that underpins it (based on the concepts and attributes defined in the Tbox). Figure 4 shows the conceptually aligned data structure generated for the sample sentence 'a woman lost her bag' discussed in previous sections. The dotted lines represent the pointers connecting the references to the woman (Abox instance e_2) in the derived tree back to the Atree and from there back to the Abox and finally the Tbox. The data structure also contains pointers between the other nodes and edges in the structure, as well as feature structures attached to each pointer, but these are not shown in the diagram. For readability the Tbox is represented here in ProFIT format (Erbach, 1995).

The sentence planner is defined as a set of mappings from pairs of Tbox concepts and constraints onto a subcategorisation frame and TAG tree for a given language, along with arguments as appropriate, such as the following example:

```
lose vt Tnx0Vnx1 perdere agent,patient
```

In this example from the Italian lexicon, the Tbox concept *lose* is mapped onto a frame that represents a transitive verb clause, *vt*, which will be realized using the XTAG (Bleam et al., 2001) initial tree *Tnx0Vnx1* and has lexical root *perdere*. The signature of the frame is defined in XML and can be reused for many different target languages:

```
<clauseFrame key="vt">
<arg1 role="subj" constraint="nominal"/>
<arg2 role="dobj" constraint="nominal"/>
</clauseFrame>
```

The frame signature defines two subsidiary arguments, each of which is annotated with a semantic role and constrained to a syntactic type. The mapping shown above maps these two arguments onto the *agent* and *patient* attributes of the concept *lose* respectively. If an Abox node that is an instance of *lose* is represented by this subcategorisation frame, then the Abox node referenced by its out-going *agent* edge will be represented by a new subcategorisation frame which: (a) is mapped to *its* underlying TBox concept; (b) fulfils the constraint that its syntactic type is *nominal*, and (c) declares the appropriate number of subsidiary arguments. In the case of the Abox presented above this number is zero, since the agent has no subsidiary attributes defined.

To build the complete Atree the system begins at the root node of the Abox and traverses the directed graph iteratively, linking the nodes and the edges of the Atree back onto the nodes and edges of the Abox as they are constructed and annotating each one both with the subcategorization frame used to instantiate it and with the TAG elementary tree which will realize it. Where an Abox instance has already been realised because of a cycle in the graph, such as node e_2 in the example Abox, the system breaks out of the current iteration and reprocesses only the repeated node itself but none of its dependents[1].

The resulting Atree structure then serves as a derivation tree from which a derived tree is constructed following the TAG formalism. This is a three step process: first the system starts from the root of the Atree and composes a syntactic tree using the elementary trees specified in each Atree node; next it inflects the lexical items in the tree based on feature structures in the Abox defining number and time period, lexical gender defined in the lexicon, and case defined in the elementary trees[2]; finally the system applies further morphological changes (such as changing *de le* to *du* for French) and orthographic changes (such as changing *a* to *an* in front

[1]The current version of the system realises first mentions using indefinite articles and determiners and subsequent mentions using definite articles or personal pronouns. We are currently enhancing the system by incorporating an existing referring expression generator into the architecture, and we intend to report that research separately.

[2]Some inflections require auxiliary trees to be adjoined to handle articles, modal auxiliaries, reflexive pronouns and so on.

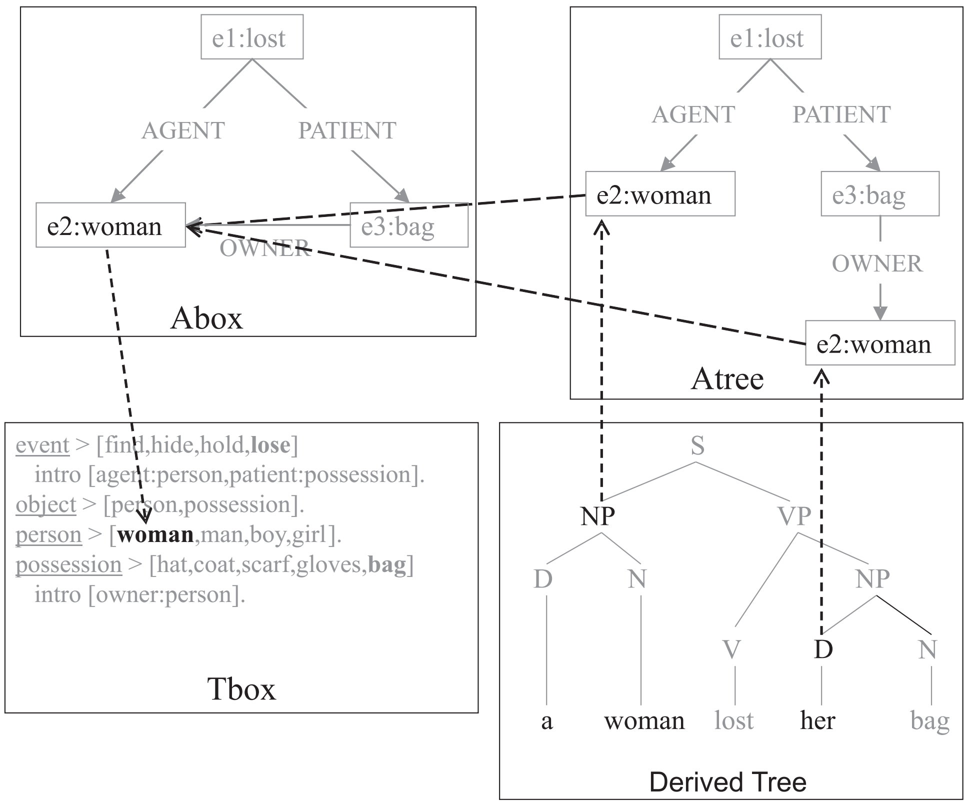

e1:lost
AGENT
PATIENT
e2:woman
e3:bag
OWNER
Abox
e1:lost
AGENT
PATIENT
e2:woman
e3:bag
OWNER
e2:woman
Atree
event > [find,hide,hold,lose]
intro [agent:person,patient:possession].
object > [person,possession].
person > [woman,man,boy,girl].
possession > [hat,coat,scarf,gloves,bag]
intro [owner:person].
Tbox
S
NP
VP
D
N
NP
V
D
N
a
woman
lost
her
bag
Derived Tree

Figure 4: The conceptually aligned text data structure

of a vowel in English) following a left-right traversal of the leaves of the tree.

3 Advantages of the alignment model

This section lists important practical benefits that result from the design and structure of the Atree, the monotonic nature of the generation algorithm, and the introduction of back-referencing pointers that preserve each stage of the generation process in memory.

3.1 Text as a function of content

There is a unique mapping from an Abox (under a given Tbox) to an Atree and corresponding derived tree for a given (controlled) natural language. As a result, the entire composite data structure can be serialised dependably as a set of triples representing the Abox; in other words, we can store the text, its linguistic structure, its semantic structure, its conceptual content and their interrelationships just by storing the Abox. At any time the complete structure can be restored by regenerating the other components of the composite data structure.

3.2 Completeness can be proved

We can prove that the generator is *complete*, in the sense that it will produce a conceptually aligned text (in the relevant controlled language) for any Abox that conforms to the Tbox. To prove completeness automatically we first check that there is a subcategorisation frame (and associated TAG elementary tree) in the grammar for each concept that can root an Abox, with the right argument structure. We then consider each attribute/subsidiary argument pair and ensure that suitable elementary trees are defined for all concepts subsumed by the attribute constraint in the TBox. This checking algorithm is performed recursively and bottoms out at the leaf nodes of the TBox. Although the set of Aboxes derivable from the TBox may be infinite, the fact that realisation is based on local information, because of the simplifying asumption of controlled natural language output, allows us to prove that they can all be realised with a finite set of checks. Note that we assume that the system is equipped with the requisite morphological and orthographic rules to realise the resulting derivation tree.

Of course, such a proof does not guarantee that the generator is *consistent* in the sense that it produces different texts for different Aboxes; for now we leave this as an open research question.

3.3 Simplified surface realisation

As others have pointed out (Candito and Kahane, 1998), a semantic graph (like the SemS in Meaning-Text Theory) bears a close analogy to the derivation tree in TAG. Replacing the semantic graph by an Atree strengthens this analogy, with the result that a derivation tree can be obtained by labelling each vertex of the Atree with a TAG elementary tree. This allows an extremely simple and efficient realisation algorithm, suitable for contexts like WYSIWYM editing where text in a controlled language must be generated almost instantaneously.

3.4 Fast scalable performance

The system is therefore fast and scales linearly, which means that it can be fielded in a context where it is required to operate in real time, even with a large Abox.

In testing, the average boot time required to load the conceptual and linguistic resources for a simple patient narrative containing 6 sentences was 42.7ms, and the average time taken to regenerate the composite data structure from a serialization of the Abox was 4.5ms, averaged over 20 iterations. The time taken to regenerate the composite data structure from a serialization of the Abox rose to 96.1ms when the Abox was increased to 600 sentences. This scaling factor is sub-linear because of caching.

3.5 Fully sufficient data structure

Since the data structure is self-sufficient there is no need for any enclosing structure to mediate access to the information encapsulated within it. For example, a WYSIWYM application based on this abstraction renders the text directly from the derived tree and uses the underlying structure to provide appropriate controls to the user so that s/he can make changes to the Abox using the text as a visualizing control. In contrast, previous versions of the WYSIWYM system described in the literature rely on a complex, bespoke *feedback text* data structure which encodes the interaction between spans of text and nodes or edges of the Abox in order to implement the same task.

3.6 Uses standard formalisms

It is important to note that while the composite data structure is itself a bespoke structure, and the Atree model is novel in design and implementation, the meaning and the text are both represented using well-known and well-understood formalisms supported by proven technologies. This means that the combined data structure forms a realistic platform for research into language generation and understanding, and also provides a reusable, practical mechanism for engineering conceptually annotated texts that are machine- and human-readable.

3.7 Supports GRE

The Atree represents precisely those features of the prior discourse that are relevant for determining the content of referring expressions: whether the referent has been mentioned before, whether it is still salient, which distractors have also been mentioned, and so forth. This suggests a two-stage process for generating referring expressions: first, derive an Atree in which all subsequent mentions of an entity are left as unlabelled terminal vertices; second, for each of these vertices, apply a content-determination algorithm to obtain an Abox fragment containing the (familiar) information that identifies the referent, and transform this to an Atree fragment rooted in the referent. In this way we obtain a convenient interface with GRE (Generation of Referring Expressions) modules which deliver a graph as output (Krahmer et al., 2003).

4 Exploitation

These properties of the algorithm and data structure can be exploited to provide extended functionality for the Semantic Web, including the services listed below.

4.1 Coreference aware text

Spans of the text which corefer share pointers to the same Abox node. This feature of the data structure allows us to add visualization controls to the text to clarify referring expressions for the reader, and also allows system processes which work with the document content to reliably and automatically identify which segments of the document text communicate which elements of content.

4.2 Semantic search

A further benefit arising from conceptual alignment is that the data structure supports semantic search of the text as a computationally trivial operation, either based on instances (encoded in the Abox) or based on concepts (encoded in the Tbox). Because the meaning of the text is reliably encoded in the Abox and Tbox no text processing is required to perform the search, and because those structures are connected back to the text any user interface application can annotate the text to display the results of the semantic search without altering or extending the composite data structure.

4.3 Query construction

In the context of semantic search the data structure also supports query construction using WYSIWYM (Hallett et al., 2007). The query interface would necessitate some small extensions to the Tbox and the linguistic resources to allow the user to construct questions, but no further work would be required to support the expression of the question content (since the data structure representing the content to be searched already supports it) nor the dynamic interaction between the text and the conceptual representation of the questions, since both are already provided by the composite data structure and its supporting algorithms.

4.4 Reformulation

Since the system has complete control of the text representing the information content of any conceptully aligned document, it can reformulate the content using any natural language for which appropriate linguistic resources have been defined. It can also make use of a range of NLG technologies to reformulate sections of the document (e.g., expanding referring expressions which the user has signalled as ambiguous) or even to reformulate whole documents, perhaps generating a summary of each document in the results page of a search.

4.5 Machine processing

While a human end user can interact with the text in a number of ways, and in a number of languages, machine processes can reason over the assertions defined by the Abox and Tbox by interacting with the same data structure. Such machine processes could include consistency or redundancy checks based on logical inferencing.

5 Conclusion

We have proposed a method for systematically aligning meaning (encoded as an Abox) with text, through the mediation of an Atree which serves both as a notational variant of the Abox, and as an abstract representation of the span structure of the text covering left-to-right order as well as dependencies. We think the Atree has some theoretical interest as an intermediate representation in NLG, but we propose it here as a practical technique for generating interactive texts in a controlled language, suitable for WYSIWYM editing and for other applications offering text-based semantic services.

The long-term tendency of this research is to support the aims of the Semantic Web by producing documents that are aligned fully with metadata encoding their meanings. By ‘full’ alignment we mean firstly that the metadata embrace the *whole* meaning of the text, not just fragments, and secondly that the alignment is fine-grained, so that every entity or relationship in the metadata is linked to the spans that express it. Such a document could be encoded by a source file containing only metadata; as well as referencing domain ontologies in the usual way, this file would reference linguistic resources for expressing the metadata in a controlled language; ideally these resources would also be defined in web ontologies, as envisioned in the GOLD project (Wilcock, 2007).

We do not of course expect that conceptual alignment will be used ubiquitously on the web. Often authors wish to retain control over wording, or are simply not interested in offering semantic services, especially when this would require a further authoring effort. However, there are many scientific, administrative or commercial contexts where conceptual alignment would yield clear benefits. One obvious candidate is the genre of instructions, which has been addressed in earlier projects on software manuals (Paris et al., 1995) and medical information leaflets (Bouayad-Agha et al., 2002). Instruction manuals are notoriously complex and difficult to maintain; for products distributed world-wide they also invite misunderstanding through poor translation or wording unsuitable for non-native speakers (the controlled language AECMA has become a standard in the aircraft industry for exactly these reasons (Wojcik and Hoard, 1995)). For this kind of document, we see broadly three potential benefits from conceptual alignment, as described in the previous section: flexible presentation (including multilinguality), enhanced navigation (semantic search), and inference-based services (including querying and consistency checking).

References

John Bateman, Robert Kasper, Johanna Moore, and Richard Whitney. 1989. A general organization of knowledge for natural language processing: The Penman Upper Model. Technical report, Information Sciences Institute, Marina del Rey, California.

T. Berners-Lee, J. Hendler, and O. Lassila. 2001. The semantic web. *Scientific American*, 284(5):34–43.

Tonia Bleam, Chung hye Han, Rashmi Prasad, Carlos Prolo, and Anoop Sarkar. 2001. A Lexicalized Tree-Adjoining Grammar for English. Technical report, The XTAG Research Group, Institute for Research in Cognitive Science, University of Pennsylvania, Philadelphia, US.

Nadjet Bouayad-Agha, Richard Power, Donia Scott, and Anja Belz. 2002. PILLS: Multilingual generation of medical information documents with overlapping content. In *Proceedings of the Third International Conference on Language Resoures and Evaluation (LREC 2002)*, pages 2111–2114, Las Palmas.

M. Candito and S. Kahane. 1998. Can the TAG derivation tree represent a semantic graph? An answer in the light of the Meaning-Text Theory. In *Proceedings of the Fourth Workshop on Tree-Adjoining Grammars and Related Frameworks*, Philadephia, USA.

G. Erbach. 1995. Profit: Prolog with features, inheritance and templates. In *Seventh Conference of the European Chapter of the Association for Computational Linguistics*, pages 180–187, Dublin.

Catalina Hallett, Donia Scott, and Richard Power. 2007. Composing queries through conceptual authoring. *Computational Linguistics*, 33(1):105–133.

Emiel Krahmer, Sebastiaan van Erk, and Andr Verleg. 2003. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72.

Cécile Paris, Keith Vander Linden, Markus Fischer, Anthony Hartley, Lyn Pemberton, Richard Power, and Donia Scott. 1995. A support tool for writing multilingual instructions. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1398–1404, Montreal, Canada.

W. K. Percival. 1990. Reflections on the history of dependency notions in linguistics. *Historiographia Linguistica*, 17(2):29–47.

R. Power and D. Scott. 1998. Multilingual authoring using feedback texts. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics*, pages 1053–1059, Montreal, Canada.

Richard Power, Donia Scott, and Nadjet Bouayad-Agha. 2003. Document structure. *Computational Linguistics*, 29(4):211–260.

G. Wilcock. 2007. An OWL Ontology for HPSG. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 169–172, Prague.

R. Wojcik and J. Hoard. 1995. Controlled Languages in Industry. In *Survey of the State of the Art in Human Language Technology*, pages 274–276.