Pacific Association for Computational Linguistics

XML BASED MULTILINGUAL AUTHORING

Guy Lapalme†, Caroline Brun‡ and Marc Dymetman‡
†*RALI-DIRO, Université de Montréal, CP. 6128, Succ Centre-Ville, Montréal, Québec, Canada, H3C 3J7* lapalme@iro.umontreal.ca
‡*Xerox Research Centre Europe, 6 chemin de Maupertuis, 38240 Meylan, France*
{Caroline.Brun,Marc.Dymetman}@xrce.xerox.com

We describe an XML implementation of a multilingual authoring system. Using this system, an author can interactively write a text conforming to well-formation content and realization rules described by an XML Schema. We show an example of an application for a class of pharmaceutical documents. Some comparisons with natural language generation systems are done.

Key words: Authoring, Multilingual, Text Generation, XML

1. AUTHORING

Modern document production integrates computer tools especially in areas where there are requirements on the content or the form of the publication, for example, for technical documentation or instruction manuals. Formatting requirements are now well understood and current text processors provide tools such as formatting styles or document templates for helping in this area. This approach relies on the fact that the author will assign some *semantics* on parts of the text: for example, a paragraph is a title, a sub-title, an example etc. Once formatting is defined for each type of text, it can be applied systematically to all occurrences in the document. This approach can also be implemented by adding tags (e.g. SGML, HTML or XML) to parts of the texts which are interpreted when the document is rendered either on the screen or on paper.

But authoring is much more than formatting and researchers have developed methods for helping an author to produce the text itself in situations where the documents belong to a restricted domain with predictable textual patterns. Some standards put very strong requirements on the information that must be provided and, in some cases, even the form of the sentences can be identified by means of controlled languages.

Over the last few years, authors of this article at Xerox Research Centre Europe have been working on a system called *Multilingual Document Authoring* (MDA) [Brun et al.2000, Dymetman et al.2000, Brun and Dymetman2002, Brun et al.2002] in which the knowledge about what constitutes a valid document is provided by means of grammars. For example, in the pharmaceutical domain, the grammars can include both world knowledge (the fact that a pharmaceutical product is given in a certain form restricts both its packaging and its way of administration) and constraints about document organization (drug form and way of administration should appear in specific subsections of the document).

MDA has been successfully applied to produce pharmaceutical information leaflets and description of biological experiment results [Brun et al.2002]. The MDA system has strong links with XML in that both view the document as a mixture of tagged tree *structures* and *surface* elements of free texts.

The MDA project is based on a an approach where the surface decisions disappear from the document to be handled exclusively by rendering mechanisms. Authoring is based on language-neutral semantic decisions to build language neutral structures. Multilingual textual output is then derived from these structures by language-specific realization mechanisms. The author can thus be guided by the specification of the document content in much the same way that a DTD or a Schema can do for the structure of a XML document. But as

we will show in section 3, DTD and Schemas are severely limited in the kind of semantic or grammatical dependencies they can express between subtrees of the document structures. Current styling tools such as XSLT or CSS were designed for layout transformations but are poorly adapted to linguistic processing.

The principles for the MDA-XML authoring system described in this paper have been implemented in a user-interface that allows a real-time interaction with a multilingual grammar. The prototype has currently two parallel grammars for French and English for pharmaceutical information notices.

The user-interface (Figure 1) is composed of text windows (one for each language) on the top of each other with a window on their left that displays the semantic tree currently being built by the interaction in the text windows. This window is shown here only for development purposes and would not normally be used in a production system. The user interacts in the language text windows by clicking on a word: if it is a non-terminal displayed in blue (see the word *form* over the menu in figure 1), a menu of acceptable choices appears from which the user selects one or backtracks to a previous choice. Once a choice is made in one language (*cream* in figure 1), the semantic tree is updated to reflect this choice: *pharmForm* is changed to *creme* in the left of figure 2 and the corresponding word *creme* has been inserted in French. As the automatic mode is "checked", all mandatory choices that depend on the user selection are immediately performed. Figure 2 shows that *conditionnement* has been changed to *tube* which in this case is the only available choice and that the administration mode *modeE* has been also added. The corresponding choices are also made in French. Thus this is a truly multilingual authoring system.

2. XML INTEGRATION

XML based authoring tools are becoming widely used in the business community for supporting the production of technical documentation, controlling their quality and improving their reusability. From a computational linguist's point of view, there might be little which seems novel or exciting in XML representations. Still XML has a great potential as a *lingua franca* and in driving a large community of users towards authoring practices where content is becoming more and more explicit. So we have tried to achieve a better integration between the MDA system and XML. The next sections describe the current experiments with our prototype for multilingual authoring to show how far we have gone and illustrate some of the difficulties in going further.

XML which stands for *eXtended Markup Language* [Bray et al.2001] has been developed in order to facilitate the annotation of information to be shared between computer systems. It is intended to be easily generated and parsed by computer systems on diverse platforms so its format is based on character streams and not internal binary ones. Being character based, it also has the nice property of being readable and editable by humans using standard text editors.

XML is based on a uniform, simple and yet powerful model of data organization: the tree defined as either a single element or an element having other trees as its sub-elements called *children*. This is basically the same model as the one chosen for the Lisp programming language more than 40 years ago. This hierarchical model is very simple and allows a simple annotation of the data. As in Lisp and Prolog, the same tree notation is used for writing programs for transforming tree structures into other tree structures. In XML Schema [Thompson et al.2001], the same notation is used for both data and validating type information.

As has been shown by Lisp over the years, this tree notation is very general and can

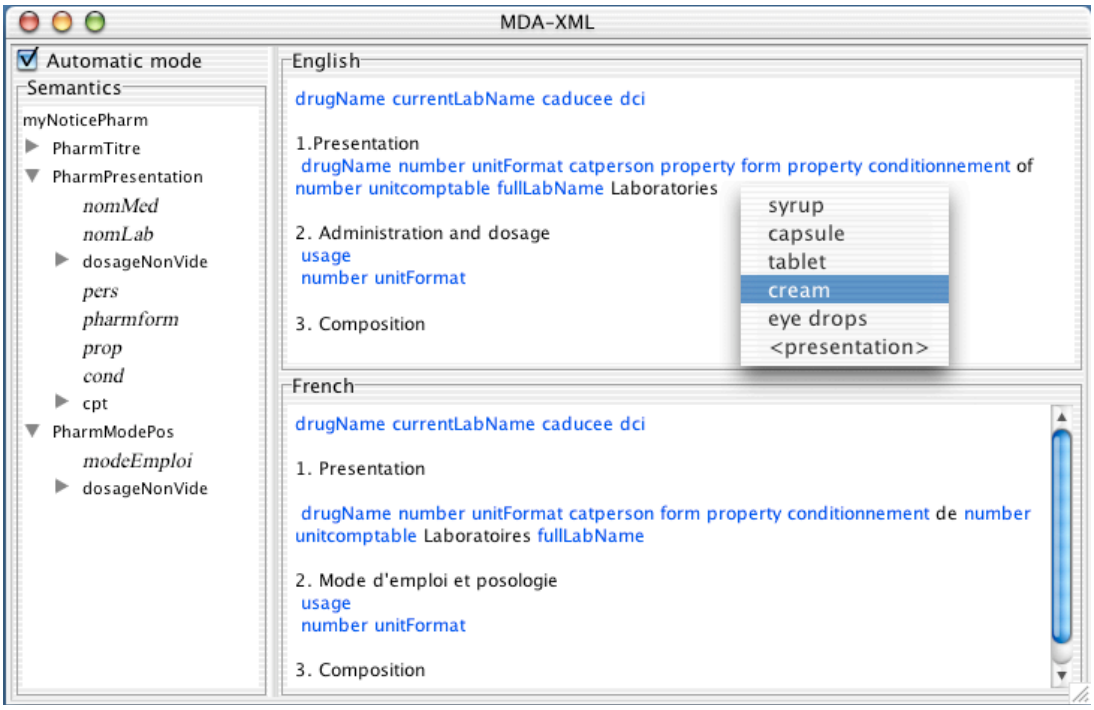

Figure 1. Screen of MDA-XML after a user has selected *form*, a menu of allowable choices is displayed.

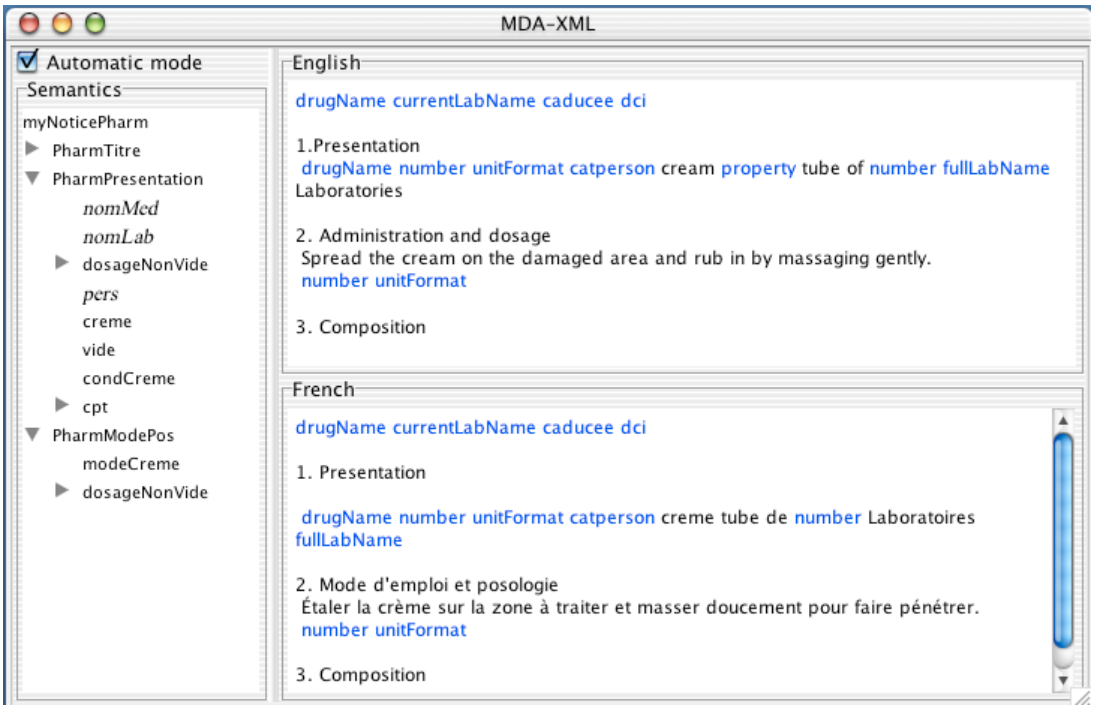

Figure 2. State of MDA-XML after the user has selected *cream* in Figure 1. Mandatory replacements have been made: *conditionnement* by *tube* and *usage* by an appropriate sentence. Semantic information has also been updated in the *Semantics* pane: *pharmform* by *creme*, *cond* by *condCreme* and *modeEmploi* by *modeCreme*.

be used not only to represent data but also its processing if one accepts that the programs are written in a similar bracketed way. Programs for transforming XML tree structures into other tree structures can be written in stylesheets (eXtensible Stylesheet Languages) which present a declarative notation for XML transformations written also in XML. Some experiments [Cawsey2000, Wilcock2001] have been done with XSLT for natural language generation that raised some problems especially in an interactive setting.

On important aspect of XML is the *a priori* type checking that can be done on the file and the validation that can be performed before processing. XML type information can be given either with a DTD (following its SGML ancestor) but recently this approach is being replaced by Schemas which offer a more powerful and flexible type system. A Schema is also written as a XML file which itself can be type checked. The growing number of both standard and custom DTD and Schemas enables the development of more and more powerful editors and processors that both implement these authoring constraints and hide the low-level XML tags from the author: a good example of this is the XMLSpy editor [Alt2002] that allows a graphic editing of Schemas, a tabular input format and a text based editing in what is dubbed the *authentic* mode.

Current XML editors stop at the presentation level. Although much work is currently being done on the *Semantic Web* or specialized XML extensions for particular applications, these works usually focus on ways of structuring and annotating data that will then the presented in various ways or processed by other computer based systems. These can be seen as textual data-base but, to our knowledge, few attempts [Boardman1999] have been made at helping authors to produce grammatical text that respects XML schemas.

In MDA, all interaction between the author and the system is done via the output text: although the choices are semantics-based, the author only sees natural language text and never deals directly with tags or semantic values which are hidden. The goal of MDA is thus to produce not a valid document in an XML sense but also grammatical text. As the internal structure of the text is always kept updated, it also propagates semantic constraints on other parts of the text and even in translations in other languages. XML rendering is thus raised to new heights with MDA.

3. DESCRIPTION OF THE PROTOTYPE: MDA-XML

As its name suggests, MDA-XML is a *rational reconstruction* of the main ideas of the original MDA system but with some different implementation choices:

- grammars are given in an XML format with a corresponding Schema instead of the current *proprietary Prolog-like* notation;
- all the interaction and interpretation is done within Java in which we implemented a *simplified unification* dealing only with atomic terms so there is no further need for an external Prolog interpreter as in the current MDA system;
- no formatting directives are implemented except for line breaks which are indicated with an HTML `<BR/>` tag.

The Java implementation follows the *classical* model-view-controller framework:

model is the semantic tree that is built interactively by the user; currently it is not possible to interact directly on the tree but only through the language panes;

view defines how the semantic tree (the model) is displayed in both the tree display and as a text in each language for a grammar which has given;

controller transmits the user input from a language pane to the model: user input can be either a mouse click to identify which node of the tree is to be expanded or a menu of allowable choices at the click displayed for the user to select an alternative that is sent back to the abstract tree (the model).

Ideally, we would have liked to implement semantic and grammar rules as XML Schema elements. Although comprehensive occurrence constraints and regular expressions can be defined for single elements, the current standard defines only limited constraint types between elements: only some relatively simple uniqueness and key-reference constraints can be enforced. Some more comprehensive types of constraint have been proposed but they rely on an external processes (e.g. XSLT) and are not appropriate for an interactive environment such as MDA-XML.

So we have defined a format for the rules which is a more or less direct translation of the standard DCG grammar rules but following XML conventions loosely patterned after the syntax used for templates definitions and calls in XSLT stylesheets. These rules can be validated by a Schema but more work should be done in order to allow a better validation at rule entry time. The Schema is also helpful in the context of an XML-aware editor which can then suggest allowable continuation at each point during rule definition entry. These rules can be entered using an XML editor such as XMLSpy which gives different views of them: textual, tabular or even a customized view.

Figure 3 gives a few examples of rules[1]: The main element is `rules` (line 1) which contain `rule` elements. A rule can call other rules by means of `call` elements. For example, at line 5, rule `noticePharm` calls the rules `callTitreNotice` (line 7) and `presentation` (line 12) embedded in litteral text and end of lines using the HTML `<BR/>` tag (lines 11,17,21). Mandatory attributes of a rule are its name, its language code (`lang` attribute) and its semantic label ("combinator" or `comb` attribute). Arguments of the combinators can be given as a list of values of attribute `arg` and should correspond to the arguments of the calls; see `arg` attribute in `call` elements, (lines 7 and 12). Similarly to what is done in XSLT, parameters of a call are given using `with-param` elements (lines 8,9,13,14,15...) within the `call` element. This is not shown in Figure 3, but rules can have an empty right hand side and they can be called recursively. Parameters can be given constant values which is quite useful for dictionary entries. A rule of the same name must be given for each language for which we want to generate the text.

As in Prolog, the links between parameters are indicated by having the same name, see line 27 where variable `NameOfMedic` used in calls `titreNotice` and `presentation`. *Logical* variables are indicated by having their names start with a capital letter or an underscore. Contrarily to Prolog and similarly to what is done in XSLT, correspondance between parameters is not done by position but by matching their attribute names. Calls and parameters of the called rules should agree. It is also important that rules be parallel between languages in order to guarantee consistent multilingual output. These validations are now performed by the MDA-XML interpreter when the grammar is loaded. But it would be preferable to define a better validation within the Schema so that the developper of an authoring system would not develop patently erroneous grammars [2] resulting in non-conforming texts.

[1] This XML format for the rules would normally not seen by the author of the rules: for example, with XMLSpy we can use the tabular form or an authentic view which completely hide the tags to only show the structure of the elements.

[2] We could also use MDA (reflectively?) for grammar authoring...

```
<rules start="noticePharm"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="mda.xsd">

   <rule name="noticePharm" lang="en" comb="myNoticePharm"
      args="titre pres mePos">
      <call name="titreNotice" arg="titre">
         <with-param name="nomMed" select="NameOfMedic"/>
         <with-param name="nomLabo" select="NameOfLabo"/>
      </call>
      <BR/><BR/>1.Presentation <BR/>
      <call name="presentation" arg="pres">
         <with-param name="typeForm" select="TYPE"/>
         <with-param name="nomMed" select="NameOfMedic"/>
         <with-param name="nomLabo" select="NameOfLabo"/>
      </call>
      <BR/><BR/>2. Dosage and administration <BR/>
      <call name="dosageUsage" arg="mePos">
         <with-param name="typeForm" select="TYPE"/>
      </call>
      <BR/><BR/>3. Composition
   </rule>

   <rule name="presentation" lang="en" comb="PharmPresentation"
         args="nomMed nomLab dos pers pharmform prop cond compt">
      <param name="typeForm" select="TYPE"/>
      <param name="nomMed" select="NameOfMedic"/>
      <param name="nomLabo" select="NameOfLabo"/>
      <call name="drugName" arg="nomMed">
         <with-param name="nomMed" select="NameOfMedic"/>
      </call>
      [...]
      <call name="form" arg="pharmform">
         <with-param name="typeForm" select="TYPE"/>
         <with-param name="num" select="sg"/>
      </call>
      [...]
   </rule>

   <rule name="usage" lang="en" comb="modeCreme" args="">
      <param name="typeForm" select="typecreme"/>

Spread the cream on the damaged area and rub in by massaging gently.
   </rule>
   [...]
</rules>
```

Figure 3. XML File showing some of the rules for authoring pharmaceutical information notices illustrated in figure 1. For the sake of saving space, only parts of rules for English are shown but parallel ones for French are also defined.

4. RELATED WORK

Authoring systems can be contrasted with natural language generation (NLG) systems which start from underlying semantic representations and produce the text without any human intervention. But some NLG systems are called *interactive* because they allow a user to dynamically modify their internal representation; after any such modification step, the text is regenerated to reflect the choice given by the author. This is specially interesting for multilingual generation because a single choice can be propagated to all languages.

An early system of this type is Drafter [Paris et al.1995] in which edition is done at the level at the semantic structure. In [Power and Scott1998], the following new idea was introduced: rather than having the user modify the semantic representation directly (which requires familiarity with the formal notation used), the generated text itself is used as interface to the semantic representation. Certain parts of the text are associated with menus which present different choices for updating the semantic representation. In this way, called by the authors WYSIWYM (What You See is What You Meant), the user never needs access to the semantic representation directly, but only indirectly, on the basis of natural language text. Coch [Coch and Chevreau2001] describes a system to interactively write weather reports in many languages; the choices in this system are more of a syntactical nature because the semantic information is extracted directly from the weather forecast system.

The MDA approach developed at XRCE in recent years is a direct follower of the Grammatical Framework (GF) [Ranta1994, Ranta2002] which is a system coming from interactive proof editors in which natural language rendering is a realization of the partial proof with hooks to proof refinements. MDA stresses GF's strong links to unification grammars and re-implements the core model in the DCG formalism.

GF-MDA thus takes as its central concept a notion of *well-formed semantic representation*. This is a tree object (called the *abstract tree*) which has to be accepted by a formal specification (a higher-order type specification in GF, a DCG in MDA) in order to be considered valid. The abstract tree is then used for generating textual realizations in different languages. The realization step uses a specific grammatical technique in GF, while MDA uses DCGs again for the realization step.

In [Power and Scott1998], the underlying semantic structure is less constrained being only described as a *knowledge base* consisting of basic predicate-argument facts. There is no notion of a specification of the well-formedness of the semantic representation. The focus is on how to interact with this semantic structure using the generated text.

The fact that MDA has a formal notion of well-formed semantic structure has important theoretical and practical advantages:

- as shown in [Dymetman et al.2000] and further expanded in this paper, MDA has a direct connection to XML theory and practice: an MDA grammar is similar to a DTD or a Schema, the abstract tree being equivalent to the XML document and realization to XML rendering;
- it makes sense to have a notion of both a well-formed abstract document, and a well-formed textual document (a realization of the latter);
- having a sound formal basis enables the characterization of some fundamental problems, such as the *life-death* of a user choice [Dymetman2002]: determining which choices of the user lead to a clash-free document only makes sense in a formal framework; this is an problem that appears as soon as dependencies are added to a context-free skeleton.

5. EXPERIMENT AND ASSESSMENT

MDA-XML has been used for creating a comprehensive bilingual grammar for pharmaceutical notices (about 2 500 lines of XML text). In particular, we were able to capitalize on industry standard XML tools: for example, we used the tabular and *authentic* views in the XMLSpy editor in order to simplify entry by having Schema dependent run-time suggestions for defining the grammar. XML tags are also completely hidden in this mode.

Contrarily to the current MDA system, MDA-XML is uniformly implemented in Java and does not depend on communicating with an external Prolog interpreter to compute its suggestions and update its display, so it is both portable and fast. There are unfortunately some drawbacks because, in some cases, having the power of a Prolog interpreter at hand can simplify the grammars by making use of non-determinism. As the current MDA-XML does not implement full unification, some constructions such as dependent lists of elements are more awkward to specify. The current MDA system also allows the use of HTML code for rendering the output but MDA-XML does not; this would be relatively simple to add though. At the current stage of development, it is too soon to evaluate to what extent MDA-XML improves the research ideas of the original MDA but the fact that MDA-XML is built upon more widely used tools will enable us to make it more widely known and accepted by users.

As in MDA, we rely on a text based interaction with the user because we think this is the most natural way. It would be interesting to see how the semantic tree could also be used for some selections. Although it is technically simple to display semantic choices in a menu in the tree view, more user interface experiments would be necessary to determine how easy it is for a user to be aware of the consequences a choice would have on the output in all languages.

Because of the relatively simple checks that can be expressed in the current Schema formalism, the grammars in MDA-XML have to be checked when they are parsed for completeness and semantical validity. It would be interesting to explore extended validation formalisms to see if some of these checks could be integrated in the definition of Schemas.

6. CONCLUSION

This paper has described MDA-XML, a multilingual authoring prototype that fits well into an XML framework. The rules it implements are written in XML and can be seen as a linguistically expressive variant of the XML Schemas already used in some authoring contexts for formatting and rendering purposes. The system has been uniformly implemented in Java (parser, grammar interpreter and graphical user interface) and has been used on small-scale pharmaceutical documents.

ACKNOWLEDGMENTS

REFERENCES

[Alt2002] Altova Corp., 2002. *XML Spy 5 Entreprise Edition Manual.*

[Boardman1999] R. Boardman. 1999. An XML/XSL architecture for language-neutral document authoring. Master's thesis, Centre for Cognitive Science, Edinburgh University.

[Bray et al.2001] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler. 2001. Extensible Markup Language (XML) 1.0 (Second Edition). Technical report, W3C.

[Brun and Dymetman2002] Caroline Brun and Marc Dymetman, 2002. *Multilinguisme et traitement de l'information*, chapter Rédaction multilingue assistée dans le modèle MDA, pages 129–152. Traité des sciences et techniques de l'information. Hermès.

[Brun et al.2000] Caroline Brun, Marc Dymetman, and Veronika Lux. 2000. Document structure and multilingual authoring. In *Proceedings of the First International Natural Language Generation Conference (INLG'2000)*, pages 24–31, Mitzpe Ramon, Israel, June.

[Brun et al.2002] Caroline Brun, Marc Dymetman, Eric Fanchon, and Stanislas L'Homme. 2002. Controlled authoring of biological experiment reports. In *submitted to EACL'03 demo session*, page 4 pages.

[Cawsey2000] Alison Cawsey. 2000. Presenting tailored resource descriptions: Will xslt do the job? In *9th International World Wide Web Conference*, May.

[Coch and Chevreau2001] Jose Coch and Karine Chevreau. 2001. Interactive multilingual generation. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*. Springer.

[Dymetman et al.2000] M. Dymetman, V. Lux, and A. Ranta. 2000. Xml and multilingual document authoring: converging trend. In *Proceedings of the The 18th International Conference on Computational Linguistics (COLING 2000)*, pages 243–249, Saarbruecken. COLING.

[Dymetman2002] Marc Dymetman. 2002. Text authoring, knowledge acquisition and description logics. In *Proceedings of Coling 2002*, Taiwan.

[Paris et al.1995] Cecile Paris, Keith Vander Linden, Markus Fisher, Anthony Hartley, Lyn Pemberton, Richard Power, and Donia Scott. 1995. A support tool for writing multilingual instructions. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1398–1404, Montréal.

[Power and Scott1998] Richard Power and Donia Scott. 1998. Multilingual authoring using feedback texts. In *Coling-ACL*, pages 1053–1059, Montréal.

[Ranta1994] Aarne Ranta. 1994. *Type Theoretical Grammar*. Oxford University Press, Oxford.

[Ranta2002] Arnte Ranta. 2002. Grammatical framework. a type-theoretical grammar formalism. *to appear in Journal of Functional Programming*.

[Thompson et al.2001] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn. 2001. XML Schema Part 1: Structures. Technical report, W3C.

[Wilcock2001] Graham Wilcock. 2001. Pipelines, templates and transformations: XML and natural language generation. In *Proceedings of the first XML and NLP workshop*, pages 1–8.