A Framework for MT and Multilingual NLG Systems Based on Uniform Lexico-Structural Processing

Benoit Lavoie
CoGenTex, Inc.
840 Hanshaw Road
Ithaca, NY
USA, 14850
benoit@cogentex.com

Richard Kittredge
CoGenTex, Inc.
840 Hanshaw Road
Ithaca, NY
USA, 14850
richard@cogentex.com

Tanya Korelsky
CoGenTex, Inc.
840 Hanshaw Road
Ithaca, NY
USA, 14850
tanya@cogentex.com

Owen Rambow *
ATT Labs-Research, B233
180 Park Ave, PO Box 971
Florham Park, NJ
USA, 07932
rambow@research.att.com

Abstract

In this paper we describe an implemented framework for developing monolingual or multilingual natural language generation (NLG) applications and machine translation (MT) applications. The framework demonstrates a uniform approach to generation and transfer based on declarative lexico-structural transformations of dependency structures of syntactic or conceptual levels ("uniform lexico-structural processing"). We describe how this framework has been used in practical NLG and MT applications, and report the lessons learned.

1 Introduction

In this paper we present a linguistically motivated framework for uniform lexico-structural processing. It has been used for transformations of conceptual and syntactic structures during generation in monolingual and multilingual natural language generation (NLG) and for transfer in machine translation (MT). Our work extends directions taken in systems such as Ariane (Vauquois and Boitet, 1985), FoG (Kittredge and Polguère, 1991), JOYCE (Rambow and Korelsky, 1992), and LFS (Iordanskaja et al., 1992). Although it adopts the general principles found in the above-mentioned systems, the approach presented in this paper is more practical, and we believe, would eventually integrate better with emerging statistics-based approaches to MT.

The framework consists of a portable Java environment for building NLG or MT applications by defining modules using a core tree transduction engine and single declarative ASCII specification language for conceptual or syntactic dependency tree structures[1] and their transformations. Developers can define new modules, add or remove modules, or modify their connections. Because the processing of the transformation engine is restricted to transduction of trees, it is computationally efficient.

Having declarative rules facilitates their reuse when migrating from one programming environment to another; if the rules are based on functions specific to a programming language, the implementation of these functions might no longer be available in a different environment. In addition, having all lexical information and all rules represented declaratively makes it relatively easy to integrate into the framework techniques for generating some of the rules automatically, for example using corpus-based methods. The declarative form of transformations makes it easier to process them, compare them, and cluster them to achieve proper classification and ordering.

* The work performed on the framework by this co-author was done while at CoGenTex, Inc.

[1] In this paper, we use the term *syntactic dependency (tree) structure* as defined in the Meaning-Text Theory (MTT; Mel'cuk, 1988). However, we extrapolate from this theory when we use the term conceptual dependency (tree) structure, which has no equivalent in MTT (and is unrelated to Shank's CD structures proposed in the 1970s).

Thus, the framework represents a generalized processing environment that can be reused in different types of natural language processing (NLP) applications. So far the framework has been used successfully to build a wide variety of NLG and MT applications in several limited domains (meteorology, battlefield messages, object modeling) and for different languages (English, French, Arabic, and Korean).

In the next sections, we present the design of the core tree transduction module (Section 2), describe the representations that it uses (Section 3) and the linguistic resources (Section 4). We then discuss the processing performed by the tree transduction module (Section 5) and its instantiation for different applications (Section 6). Finally, we discuss lessons learned from developing and using the framework (Section 7) and describe the history of the framework comparing it to other systems (Section 8).

2 The Framework's Tree Transduction Module

The core processing engine of the framework is a generic tree transduction module for lexico-structural processing, shown in Figure 1. The module has dependency stuctures as input and output, expressed in the same tree formalism, although not necessarily at the same level (see Section 3). This design facilitates the pipelining of modules for stratificational transformation. In fact, in an application, there are usually several instantiations of this module.

The transduction module consists of three processing steps: lexico-structural pre-processing, main lexico-structural processing, and lexico-structural post-processing. Each of these steps is driven by a separate grammar, and all three steps draw on a common feature data base and lexicon. The grammars, the lexicon and the feature data base are referred to as the *linguistic resources* (even if they sometimes apply to a conceptual representation). All linguistic resources are represented in a declarative manner. An *instantiation* of the tree transduction module consists of a specification of the linguistic resources.

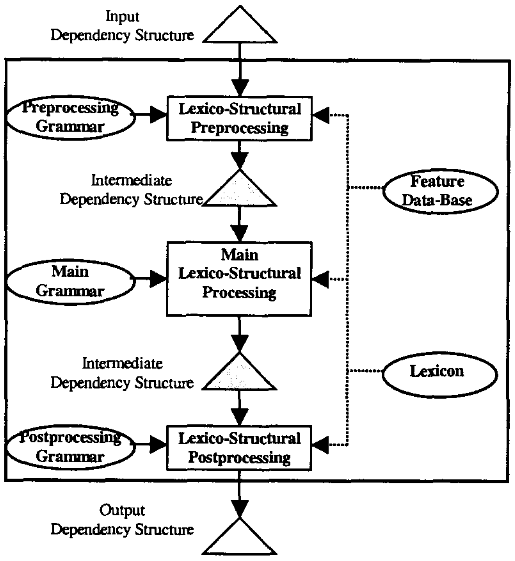


Figure 1: Design of the Tree Transduction Module

3 The Framework's Representations

The representations used by all instantiations of the tree transduction module in the framework are dependency tree structures. The main characteristics of all the dependency tree structures are:

- A dependency tree is unordered (in contrast with phrase structure trees, there is no ordering between the branches of the tree).
- All the nodes in the tree correspond to lexemes (i.e., lexical heads) or concepts depending on the level of representation. In contrast with a phrase structure representation, there are no phrase-structure nodes labeled with nonterminal symbols. Labelled arcs indicate the dependency relationships between the lexemes.

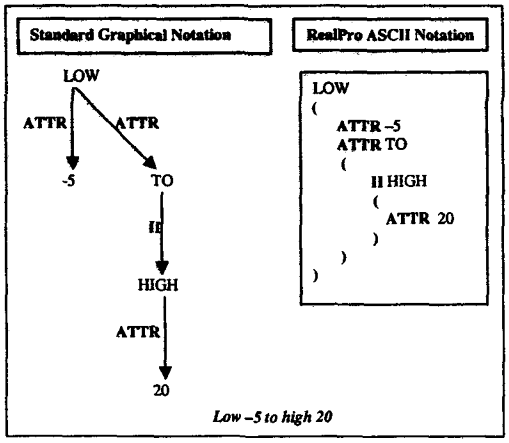
The first of these characteristics makes a dependency tree structure a very useful representation for MT and multilingual NLG, since it gives linguists a representation that allows them to abstract over numerous cross-linguistic divergences due to language specific ordering (Polguère, 1991).

We have implemented 4 different types of dependency tree structures that can be used for NLG, MT or both:

- Deep-syntactic structures (DSyntSs);
- Surface syntactic structures (SSyntSs);

- Conceptual structures (ConcSs);
- Parsed syntactic structures (PSyntSs).

The DSyntSs and SSyntSs correspond closely to the equivalent structures of the Meaning-Text Theory (MTT; Mel'cuk, 1988): both structures are unordered syntactic representations, but a DSyntS only includes full meaning-bearing lexemes while a SSyntS also contains function words such as determiners, auxiliaries, and strongly governed prepositions. In the implemented applications, the DSyntSs are the pivotal representations involved in most transformations, as this is also often the case in practice in linguistic-based MT (Hutchins and Somers, 1997). Figure 2 illustrates a DSyntS from a meteorological application, MeteoCogent (Kittredge and Lavoie, 1998), represented using the standard graphical notation and also the RealPro ASCII notation used internally in the framework (Lavoie and Rambow, 1997). As Figure 2 illustrates, there is a straightforward mapping between the graphical notation and the ASCII notation supported in the framework. This also applies for all the transformation rules in the framework which illustrates the declarative nature of our approach.



```
LOW
(
   ATTR –5
   ATTR TO
   (
      II HIGH
      (
        ATTR  20
      )
   )
)
```

Figure 2: DSyntS (Graphical and ASCII Notation)

The ConcSs correspond to the standard frame-like structures used in knowledge representation, with labeled arcs corresponding to slots. We have used them only for a very limited meteorological domain (in MeteoCogent), and we imagine that they will typically be defined in a domain-specific manner.

Figure 3 illustrates the mapping between an interlingua defined as a ConcS and a corresponding English DSyntS. This example, also taken from MeteoCogent, illustrates that the conceptual interlingua in NLG can be closer to a database representation of domain data than to its linguistic representations.

As mentioned in (Polguère, 1991), the high level of abstraction of the ConcSs makes them a suitable interlingua for multilingual NLG since they bridge the semantic discrepancies between languages, and they can be produced easily from the domain data. However, most off-the-shelf parsers available for MT produce only syntactic structures, thus the DSyntS level is often more suitable for transfer.

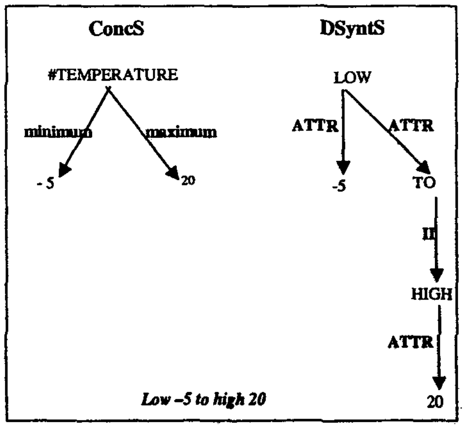


Figure 3: ConcS Interlingua and English DSyntS

Finally, the PSyntSs correspond to the parser outputs represented using RealPro's dependency structure formalism. The PSyntSs may not be valid directly for realization or transfer since they may contain unsupported features or dependency relations. However, the PSyntSs are represented in a way to allow the framework to convert them into valid DSyntS via lexico-structural processing. This conversion is done via conversion grammars customized for each parser. There is a practical need to convert one syntactic formalism to another and so far we have implemented converters for three off-the-shelf parsers (Palmer et al., 1998).

4 The Framework's Linguistic Resources

As mentioned previously, the framework is composed of instantiations of the tree transduction module shown in Figure 1. Each module has the following resources:

- *Feature Data-Base*: This consists of the feature system defining available features and their possible values in the module.
- *Lexicon*: This consists of the available lexemes or concepts, depending on whether the module works at syntactic or conceptual level. Each lexeme and concept is defined with its features, and may contain specific lexico-structural rules: transfer rules for MT, mapping rules to the next level of representation for surface realization of DSyntS or lexicalization of ConcS.
- *Main Grammar*: This consists of the lexico-structural mapping rules that apply at this level and which are not lexeme- or concept-specific (e.g. DSynt-rules for the DSynt-module, Transfer-rules for the Transfer module, etc.)
- *Preprocessing grammar*: This consists of the lexico-structural mapping rules for transforming the input structures in order to make them compliant with the main grammar, if this is necessary. Such rules are used to integrate new modules together when discrepancies in the formalism need to be fixed. This grammar can also be used for adding default features (e.g. setting the default number of nouns to singular) or for applying default transformations (e.g. replacing non meaning-bearing lexemes with features).
- *Postprocessing grammar*: This consists of lexico-structural mapping rules for transforming the output structures before they can be processed by the next module. As for the preprocessing rules, these rules can be used to fix some discrepancies between modules.

Our representation of the lexicon at the lexical level (as opposed to conceptual) is similar to the one found in RealPro. Figure 4 shows a specification for the lexeme SELL. This lexeme is defined as a verb of regular morphology with two lexical-structural mappings, the first one introducing the preposition TO for its 3rd actant, and the preposition FOR for its 4th actant: *(a seller) X1 sells (merchandise) X2 to (a buyer) X3 for (a price) X4.* What is important is that each mapping specifies a transformation between structures at different levels of representation but that are represented in one and the same representation formalism (DSyntS and SSyntS in this case). As we will see below, grammar rules are also expressed in a similar way.

```
LEXEME:       SELL
CATEGORY:     verb
FEATURES:     [ ]
GOV-PATTERN: [
               DSYNT-RULE:
                    SELL ( III $X3 )
                 <-->
                    SELL
                        ( completive2 TO
                            ( prepositional $X3 ) )

               DSYNT-RULE:
                    SELL ( IV $X4 )
                 <-->
                    SELL
                        ( completive3 FOR
                              ( prepositional $X4 ) )
                ]
MORPHOLOGY:     [
                  ( [ tense:past ]       sold  [ inv ]
                  ( [ mood:past-part ]   sold  [ inv ]
                  ( [ ]                  sell  [ reg ]
                ]
```

Figure 4: Specification of Lexeme SELL

At the conceptual level, the conceptual lexicon associates lexical-structural mapping with concepts in a similar way. Figure 5 illustrates the mapping at the deep-syntactic level associated with the concept #TEMPERATURE. Except for the slight differences in the labelling, this type of specification is similar to the one used on the lexical level. The first mapping rule corresponds to one of the lexico-structural transformations used to convert the interlingual ConcS of Figure 3 to the corresponding DSyntS.

```
CONCEPT: #TEMPERATURE
LEXICAL: [
         LEX-RULE:
            #TEMPERATURE   ( #minimum  $X
                             #maximum  $Y )
         <-->
            LOW   ( ATTR   $X
                    ATTR   TO
                    ( II HIGH
                        ( ATTR $Y ) ) )

         LEX-RULE:
            #TEMPERATURE   ( #minimum  $X )
         <-->
            LOW   ( ATTR   $X   )

         LEX-RULE:
            #TEMPERATURE   ( #maximum  $X )
         <-->
            HIGH   ( ATTR   $X   )
         ]
```

Figure 5: Specification of Concept #TEMPERATURE

Note that since each lexicon entry can have more than one lexical-structural mapping rule, the list of these rules represents a small grammar specific to this lexeme or concept.

Realization grammar rules of the main grammar include generic mapping rules (which are not lexeme-specific) such as the DSyntS-rule illustrated in Figure 6, for inserting a determiner.

```
DSYNT-RULE:

      $X [ class:noun article:def ]
<-->
      $X ( determinative THE )
```

Figure 6: Deep-Syntactic Rule for Determiner Insertion

The lexicon formalism has also been extended to implement lexeme-specific lexico-structural transfer rules. Figure 7 shows the lexico-structural transfer of the English verb lexeme MOVE to French implemented for a military and weather domain (Nasr et al., 1998):

*Cloud will **move into** the western regions.*
*→ Des nuages **envahiront** les régions ouest.*

*They **moved** the assets forward.*
*→ Ils ont **amené** les ressources vers l'avant.*

*The 79 dcg **moves** forward.*
*→ La 79 dcg **avance** vers l'avant.*

*A disturbance will **move** north of Lake Superior.*
*→ Une perturbation se **déplacera** au nord du lac supérieur.*

```
LEXEME:     MOVE
CATEGORY:   verb
FEATURES:   [ ]
TRANSFER:   [
             TRANSFER-RULE:
                  MOVE
                   ( ATTR INTO [class:preposition]
                      ( II $X1 ) )
                <-->
                  ENVAHIR  [class:verb]
                      ( II $X1 )

             TRANSFER-RULE:
                  MOVE
                   ( II $X2 )
                <-->
                  AMENER [class:verb]
                   ( II $X2 )

             TRANSFER-RULE:
                  MOVE
                   ( ATTR $X [lexeme:FORWARD class:adverb] )
                <-->
                  AVANCER
                   ( ATTR $X )

             TRANSFER-RULE:
                  MOVE
                <-->
                  DEPLACER [class:verb refl:+]
            ]
```

Figure 7: Lexico-Structural Transfer of English Lexeme MOVE to French

More general lexico-structural rules for transfer can also be implemented using our grammar rule formalism. Figure 8 gives an English-French transfer rule applied to a weather domain for the transfer of a verb modified by the adverb ALMOST:

*It **almost** rained.*
*→ Il a **failli** pleuvoir.*

```
TRANSFER-RULE:

          $X [ class:verb ]
              ( ATTR ALMOST )
<-->
          FAILLIR [ class:verb ]
              ( II $X [ mood:inf ] )
```

Figure 8: English to French Lexico-Structural Transfer Rule with Verb Modifier ALMOST

More details on how the structural divergences described in (Dorr, 1994) can be accounted for using our formalism can be found in (Nasr et al., 1998).

5 The Rule Processing

Before being processed, the rules are first compiled and indexed for optimisation. Each module applies the following processing.

The rules are assumed to be ordered from most specific to least specific. The application of the rules to the structures is top-down in a recursive way from the first rule to the last. For the main grammar, before applying a grammar rule to a given node, dictionary lookup is carried out in order to first apply the lexeme- or concept-specific rules associated with this node. These are also assumed to be ordered from the most specific to the least specific.

If a lexico-structural transformation involves switching a governor node with one of its dependents in the tree, the process is reapplied with the new node governor. When no more rules can be applied, the same process is applied to each dependent of the current governor. When all nodes have been processed, the processing is completed.

6 Using the Framework to build Applications

Figure 9 shows how different instantiations of the tree transduction module can be combined to build NLP applications. The diagram does not represent a particular system, but rather shows the kind of transformations that have been implemented using the framework, and how they interact. Each arrow represents one type of processing implemented by an instantiation of the tree transduction module. Each triangle represents a different level of representation.

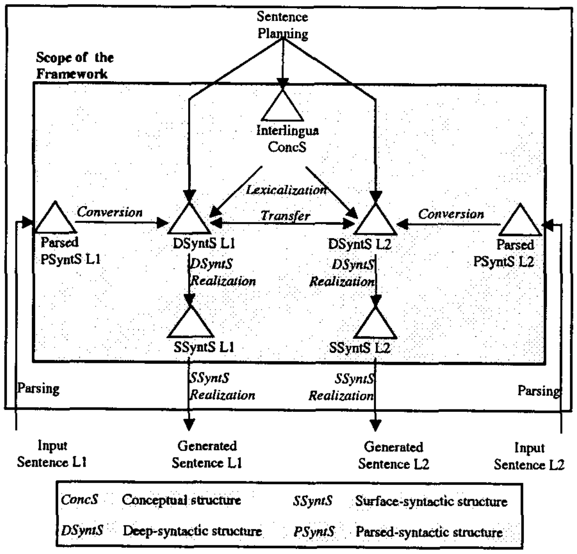


Figure 9: Scope of the Framework's Transformations

For example, in Figure 9, starting with the "Input Sentence L1" and passing through Parsing, Conversion, Transfer, DSyntS Realization and SSyntS Realization to "Generated Sentence L2" we obtain an L1-to-L2 MT system. Starting with "Sentence Planning" and passing through DSyntS Realization, and SSyntS Realization (including linearization and inflection) to "Generated Sentence L1", we obtain a monolingual NLG system for L1.

So far the framework has been used successfully for building a wide variety of applications in different domains and for different languages:

NLG:

- Realization of English DSyntSs via SSyntS level for the domains of meteorology (MeteoCogent; Kittredge and Lavoie, 1998) and object modeling (ModelExplainer; Lavoie et al., 1997).
- Generation of English text from conceptual interlingua for the meteorology domain (MeteoCogent). (The design of the interlingua can also support the generation of French but this functionality has not yet been implemented.)

MT:

- Transfer on the DSyntS level and realization via SSyntS level for English—French, English—Arabic, English—Korean and Korean—English. Translation in the meteorology and battlefield domains (Nasr et al., 1998).
- Conversion of the output structures from off-the-shelf English, French and Korean parsers to DSyntS level before their processing by the other components in the framework (Palmer et al., 1998).

7 Lessons Learned Using the Framework

Empirical results obtained from the applications listed in Section 6 have shown that the approach used in the framework is flexible enough and easily portable to new domains, new languages, and new applications. Moreover, the time spent for development was relatively short compared to that formerly required in developing similar types of applications. Finally, as intended, the limited computational power of the transduction module, as well as careful implementation, including the compilation of declarative linguistic knowledge to Java, have ensured efficient run-time behavior. For example, in the MT domain we did not originally plan for a separate conversion step from the parser output to DSyntS. However, it quickly became apparent that there was a considerable gap between the output of the parsers we were using and the DSyntS representation that was required, and furthermore, that we could use the tree transduction module to quickly bridge this gap.

Nevertheless, our tree transduction-based approach has some important limitations. In particular, the framework requires the developer of the transformation rules to maintain them and specify the order in which the rules must be applied. For a small or a stable grammar, this does not pose a problem. However, for large or rapidly changing grammar (such as a transfer grammar in MT that may need to be adjusted when switching from one parser to another), the burden of the developer's task may be quite heavy. In practice, a considerable amount of time can be spent in testing a grammar after its revision.

Another major problem is related to the maintenance of both the grammar and the lexicon. On several occasions during the development of these resources, the developer in charge of adding lexical and grammatical data must make some decisions that are domain specific. For example, in MT, writing transfer rules for terms that can have several meanings or uses, they may simplify the problem by choosing a solution based on the context found in the current corpus, which is a perfectly natural strategy. However, later, when porting the transfer resources to other domains, the chosen strategy may need to be revised because the context has changed, and other meanings or uses are found in the new corpora. Because the current approach is based on handcrafted rules, maintenance problems of this sort cannot be avoided when porting the resources to new domains.

An approach such as the one described in (Nasr et al., 1998; and Palmer and al., 1998) seems to be solving a part of the problem when it uses corpus analysis techniques for automatically creating a first draft of the lexical transfer dictionary using statistical methods. However, the remaining work is still based on handcrafting because the developer must refine the rules manually. The current framework offers no support for merging handcrafted rules with new lexical rules obtained statistically while preserving the valid handcrafted changes and deleting the invalid ones. In general, a better integration of linguistically based and statistical methods during all the development phases is greatly needed.

8 History of the Framework and Comparison with Other Systems

The framework represents a generalization of several predecessor NLG systems based on Meaning-Text Theory: FoG (Kittredge and Polguère, 1991), LFS (Iordanskaja et al., 1992), and JOYCE (Rambow and Korelsky, 1992). The framework was originally developed for the realization of deep-syntactic structures in NLG (Lavoie and Rambow, 1997). It was later extended for generation of deep-syntactic structures from conceptual interlingua (Kittredge and Lavoie, 1998). Finally, it was applied to MT for transfer between deep-syntactic structures of different languages (Palmer et al., 1998). The current framework encompasses the full spectrum of such transformations, i.e. from the processing of conceptual structures to the processing of deep-syntactic structures, either for NLG or MT.

Compared to its predecessors (Fog, LFS, JOYCE), our approach has obvious advantages in uniformity, declarativity and portability. The framework has been used in a wider variety of domains, for more languages, and for more applications (NLG as well as MT). The framework uses the same engine for all the transformations at all levels because all the syntactic and conceptual structures are represented as dependency tree structures.

In contrast, the predecessor systems were not designed to be rapidly portable. These systems used programming languages or scripts for the implementation of the transformation rules, and used different types of processing at different levels of representation. For instance, in LFS conceptual structures were represented as graphs, whereas syntactic structures were represented as trees which required different types of processing at these two levels.

Our approach also has some disadvantages compared with the systems mentioned above.

Our lexico-structural transformations are far less powerful than those expressible using an arbitrary programming language. In practice, the formalism that we are using for expressing the transformations is inadequate for long-range phenomena (inter-sentential or intra-sentential), including syntactic phenomena such as long-distance *wh*-movement and discourse phenomena such as anaphora and ellipsis. The formalism could be extended to handle intra-sentential syntactic effects, but inter-sentential discourse phenomena probably require procedural rules in order to access lexemes in other sentences. In fact, LFS and JOYCE include a specific module for elliptical structure processing.

Similarly, the limited power of the tree transformation rule formalism distinguishes the framework from other NLP frameworks based on more general processing paradigms such as unification of FUF/SURGE in the generation domain (Elhadad and Robin, 1992).

9 Status

The framework is currently being improved in order to use XML-based specifications for representing the dependency structures and the transformation rules in order to offer a more standard development environment and to facilitate the framework extension and maintenance.

Acknowledgements

A first implementation of the framework (C++ processor and ASCII formalism for expressing the lexico-structural transformation rules) applied to NLG was developed under SBIR F30602-92-C-0015 awarded by USAF Rome Laboratory. The extensions to MT were developed under SBIR DAAL01-97-C-0016 awarded by the Army Research Laboratory. The Java implementation and general improvements of the framework were developed under SBIR DAAD17-99-C-0008 awarded by the Army Research Laboratory. We are thankful to Ted Caldwell, Daryl McCullough, Alexis Nasr and Mike White for their comments and criticism on the work reported in this paper.

References

Dorr, B. J. (1994) Machine translation divergences: A formal description and proposed solution. In *Computational Linguistics*, vol. 20, no. 4, pp. 597–635.

Elhadad, M. and Robin, J. (1992) Controlling Content Realization with Functional Unification Grammars. In *Aspects of Automated Natural Language Generation*, Dale, R., Hovy, E., Rosner, D. and Stock, O. Eds., Springer Verlag, pp. 89–104.

Hutchins, W. J. and Somers, H. L. (1997) *An Introduction to Machine Translation.* Academic Press, second edition.

Iordanskaja, L., Kim, M., Kittredge, R., Lavoie, B. and Polguère, A. (1992) Generation of Extended Bilingual Statistical Reports. In *Proceedings of the 15th International Conference on Computational Linguistics*, Nantes, France, pp. 1019–1023.

Kittredge, R. and Lavoie, B. (1998) MeteoCogent: A Knowledge-Based Tool For Generating Weather Forecast Texts. In *Proceedings of the American Meteorological Society AI Conference (AMS-98),* Phoenix, Arizona, pp. 80–83.

Kittredge, R. and Polguère, A. (1991) Dependency Grammars for Bilingual Text Generation: Inside FoG's Stratificational Models. In *Proceedings of the International Conference on Current Issues in Computational Linguistics*, Penang, Malaysia, pp. 318–330.

Lavoie, B. (1995) Interlingua for Bilingual Statistical Reports. In *Notes of IJCAI-95 Workshop on Multilingual Text Generation*, Montréal, Canada, pp. 84–94 .

Lavoie, B. and Rambow, O. (1997) A Fast and Portable Realizer for Text Generation Systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, DC., pp. 265–268.

Lavoie, B., Rambow, O. and Reiter, E. (1997) Customizable Descriptions of Object-Oriented Models. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, DC., pp. 253–256.

Mel'cuk, I. (1988) Dependency Syntax. State University of New York Press, Albany, NY.

Nasr, A., Rambow, O., Palmer, M. and Rosenzweig, J. (1998) Enriching lexical transfer with cross-linguistic semantic features. In *Proceedings of the Interlingua Workshop at the MT Summit*, San Diego, California.

Palmer, M., Rambow, O. and Nasr, A. (1998) Rapid Prototyping of Domain-Specific Machine Translation Systems. *In Proceedings of the Third Conference on Machine Translation in the Americas (AMTA-98),* PA, USA, pp. 95–102.

Polguère, A. (1991) Everything has not been said about interlinguae: the case of multi-lingual text generation system. In *Proc. of Natural Language Processing Pacific Rim Symposium*, Singapore.

Rambow, O. and Korelsky, T. (1992) Applied Text Generation. In *Proceedings of the 6th International Workshop on Natural Language Generation,* Trento, Italy, pp. 40–47.

Vauquois, B. and Boitet C. (1985) Automated translation at Grenoble University. In *Computational Linguistics*, Vol. 11, pp. 28–36.