Towards a Generic Multilingual Dependency Grammar for Text Generation

François Lareau
Pompeu Fabra University

and

Leo Wanner
Pompeu Fabra University
and ICREA

Proceedings of the GEAF 2007 Workshop

Tracy Holloway King and Emily M. Bender (Editors)

CSLI Studies in Computational Linguistics ONLINE

Ann Copestake (Series Editor)

2007

CSLI Publications

http://csli-publications.stanford.edu/

Abstract

For practical multilingual text generation, efficient development and representation of large scale grammatical and lexical resources are crucial. One way to ensure efficiency is to share resources as much as possible between languages. We present some preliminary work on shared grammatical resource development within the framework of the Meaning-Text Theory, emphasizing the lexicalist point of view. We show that rich dictionaries allow for more generic grammar rules which can be used for several languages, so that the number of language-specific rules is kept low. We also discuss the benefits of shifting the workload to the dictionaries from the viewpoint of extension and consistency control as well as the impact it has on the organization of work. Furthermore, we address evaluation methodologies for the shared grammatical resources we develop.

1 Introduction

Practical multilingual natural language generation (MNLG) cannot be achieved without large scale grammatical and lexical resources. For an efficient development of multilingual broad coverage grammatical resources, two different strategies have been applied: *grammar porting* (Alshawi, 1992; Kim et al., 2003) and *grammar sharing* (Avgustinova and Uszkoreit, 2000; Bender et al., 2002; Bateman et al., 2005; Santaholma, 2007). In this article, we present some preliminary work on the development of shared grammatical MNLG resources in the framework of the dependency-based *Meaning-Text Theory*, MTT (Mel'čuk, 1988). MTT has traditionally been popular in text generation due to its multi-stratal linguistic model, which allows us, on the one hand, to select for the input structure a degree of abstraction that suits best the application in question, and, on the other hand, to keep the generation resources as modular and as simple as possible.

According to MTT, sentence generation is viewed as a sequence of transductions between structures of adjacent strata. Depending on the required degree of abstraction, generators may start from the conceptual, semantic, or syntactic structure (see also below). Each transduction is realized by a separate language-dependent grammar such that grammar developers are faced with the task of developing $n \times (m-1)$ grammars for each application (with n being the number of languages covered and m the number of strata involved in the generation process). The need for efficient sharing of grammatical resources across languages is thus obvious.

[†]The work described in this paper has been funded by the European Commission in the framework of the *eContent* Programme under the contract number EDC-11258. We would like to thank all colleagues who have contributed to the development of the resources presented here: Margarita Alonso Ramos, Bernd Bohnet, Kim Gerdes, Simon Mille, Christophe Onambele Manga, Patrycja Przewoźnik, and Vanesa Vidal. Special thanks go to Bernd Bohnet, who acted as firefighter whenever MATE was not behaving as the grammarians expected. Many thanks also to Emily Bender and Tracy Holloway King for suggestions that significantly improved the final version of the paper.

In our current application, we cover six languages (Catalan, English, French, Polish, Portuguese, and Spanish) for the domain of air quality, using as development framework and generator the graph grammar-based workbench MATE (Bohnet et al., 2000; Bohnet, 2006). It turned out that the effect of resource sharing even across languages that belong to different families (Romance, Germanic, and Slavic) is considerable. In what follows, we present our experience.

The remainder of the article is structured as follows. In Section 2, we give a short introduction to MTT. Section 3 describes the formalism used for the dictionaries and grammars in MATE. Section 4 contains the general principles that underlie our grammatical resource architecture. In Section 5, we assess the benefits of this architecture for efficient grammar development, before presenting in Section 6 an evaluation of the resources thus obtained. Section 7, finally, summarizes the central aspects of our approach and offers some conclusions.

2 Overview of MTT

As already mentioned above, MTT is based on a multi-stratal linguistic model. In total, seven different strata are distinguished, of which five are immediately relevant to written language generation: (i) the semantic stratum, (ii) the deep-syntactic stratum, (iii) the surface-syntactic stratum, (iv) the deep-morphological stratum,[1] and (v) the surface-morphological stratum. For generation applications that start from a non-linguistic content representation or even from numerical data series (as we do), an additional *conceptual* stratum is added.

Each stratum has its own alphabet over which structures for that stratum are defined, and its own interpretation for those structures. Thus, conceptual structures (ConS) are *conceptual graphs* in the sense of Sowa (2000). Semantic structures (SemS) are predicate-argument graphs with nodes labeled by semantemes and arcs labeled by the ordinal numbers of the argument relations (ordered in ascending degree of obliqueness). Deep-syntactic structures (DSyntS) are dependency trees with nodes labeled by "deep" lexical units (LUs)[2] and arcs labeled by universal syntactic relations: argument (I, II, III, ...), attributive (ATTR), and coordinative (COORD). Surface-syntactic structures (SSyntS) are dependency trees with nodes labeled by any kind of lexeme (including closed class lexemes) and arcs labeled by grammatical functions (subject, direct object, ...); SSyntS is thus equivalent to the f-structure in LFG. Deep-morphological structures (DMorphS) are chains of lemmas annotated with all relevant morpho-syntactic features. Surface-morphological structures (SMorphS) are similar to DMorphS except that contractions, elisions,

[1] In the MTT literature, the deep-morphological stratum has recently also been referred to as "Topological Stratum" (Gerdes and Kahane, 2007).

[2] The set of deep LUs of a language $\mathcal{L}$ contains all LUs of $\mathcal{L}$—with some specific additions and exclusions. Added are two types of "artificial" LUs: (i) symbols of *lexical functions* (LFs), which are used to encode lexico-semantic derivation and lexical co-occurrence (Mel'čuk, 1996); (ii) fictitious lexemes, which represent idiosyncratic syntactic constructions of $\mathcal{L}$. Excluded are: (i) structural words, (ii) substitute pronouns and values of LFs.

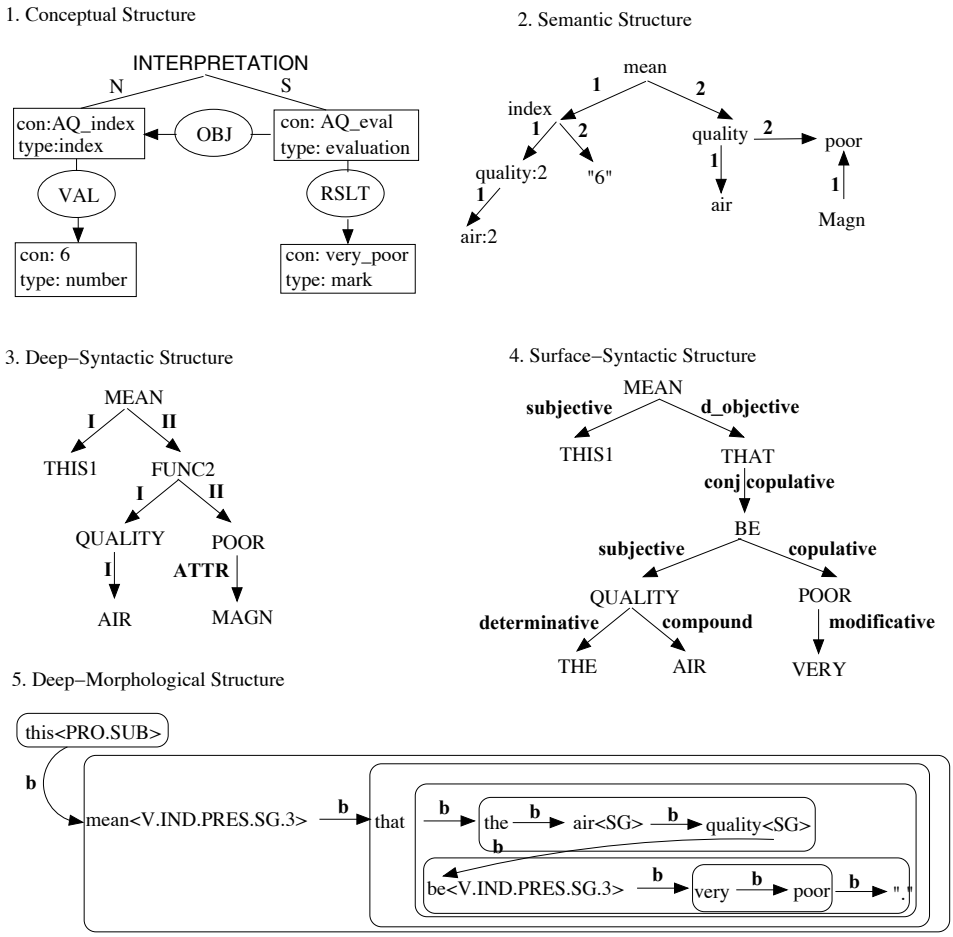

Figure 1: Sample structures at different strata of an MTT model

epenthesis and morph amalgamation have been performed. Figure 1 illustrates the first five types of structures for the sentence *This means that the air quality is very poor*; the SMorphS is obvious and does not need explicit illustration.

For each pair of adjacent strata $\mathfrak{S}_i$ and $\mathfrak{S}_{i+1}$, a separate grammar module $\mathfrak{G}^i_{i+1}$ is defined such that any well-formed structure S_{i_j} of $\mathfrak{S}_i$ can be mapped by $\mathfrak{G}^i_{i+1}$ onto a well-formed structure S_{i+1_k} of $\mathfrak{S}_{i+1}$, with S_{i_j} and S_{i+1_k} being equivalent with respect to their meaning. For convenience, we introduce a further grammar module to map a SMorphS onto a text string. As a rule, the mapping requires access to dictionaries containing information concerning the units of S_{i_j} and S_{i+1_k}.

3 Formal Framework: MATE

The MATE workbench consists of a number of support modules for the development of dictionaries and grammars and a transduction-based generator that maps

any automatically derived or manually specified input structure S_{i_j} of the stratum $\mathfrak{S}_i$ onto its equivalent structure S_{i+1_k} of the stratum $\mathfrak{S}_{i+1}$ by applying the corresponding grammar module $\mathfrak{G}^i_{i+1}$ to S_{i_j} under the use of dictionaries.

3.1 Dictionary Encoding in MATE

Dictionaries contain two types of information: (i) information concerning the elements of the different node alphabets (the vocabulary) and (ii) information concerning the correspondence between elements of node alphabets of adjacent strata. Therefore, three main dictionaries are available: a conceptual dictionary, a semantic dictionary and a lexical dictionary. All are organized in terms of recursive feature structures.

The *conceptual dictionary* is used, first of all, to encode concept-semanteme mapping information; cf. a simplified entry for the concept CONCENTRATION:[3]

concentration: property_attribute {
sem = 'concentration'
MATR = {relation = 1 target=referent}
VAL = {relation = 2 target=referent}
ATTR = {relation = 1 source=referent}}

The concept CONCENTRATION has two argument slots: something which has a concentration (referred to as MATR in accordance with Sowa (2000)), and a value (referred to as VAL), i.e., an absolute concentration figure. The concept may also be modified by a qualitative characterization of the concentration ("high", "low", etc.), referred to as ATTR. The corresponding semanteme 'concentration' takes MATR as its first semantic argument (indicated by the "relation=1" parameter embedded in MATR's value) and VAL as its second. The attributes "target=referent" and "source=referent" indicate the direction of the semantic relation (for MATR and VAL, the semantic predicate is 'concentration', which takes MATR's and VAL's corresponding semantemes as its arguments, while ATTR's semantic correspondent is a predicate taking 'concentration' as its argument).

The *semantic dictionary* gives, for every semanteme described, all its possible lexicalisations. For instance, the meaning 'cause' would be mapped to the LUs CAUSE$_{[V]}$, CAUSE$_{[N]}$, RESULT$_{[V]}$, RESULT$_{[N]}$, DUE, BECAUSE, CONSEQUENCE, etc. Note that we do not consider at this stage the valency of the LUs. Thus, it does not matter that *X causes Y* means that *Y results from X*; what interests us here is only that these two lexemes can both be used to denote the same situation, regardless of the communicative orientation. Cf., for illustration, the entry for the semanteme 'concentration' as specified in the semantic dictionary:

concentration {
label = parameter
lex = concentration }

[3]More information can be added to this basic entry, but we leave it aside in this paper.

The semantic type of a semanteme can be specified in the semantic dictionary (cf. “label=parameter”). It is also possible to specify the semantic type of the arguments of a predicate. For instance, adding the attribute “1=substance” here would force the first semantic argument of ‘concentration’ to be of type “substance”.

The *lexical dictionary* contains, for each LU, at least information on its part of speech and minimal sub-categorization information. For more elaborate generation, the whole variety of sub-categorization patterns and lexical co-occurrence (i.e., *collocation*) information should also be captured. As already mentioned above (see footnote 2), the latter is specified in terms of *lexical functions* (LFs). A lexical co-occurrence LF is a directed lexico-semantic relation that holds between two LUs that form a lexically restricted expression (e.g. *heavy smoker*, *make a statement*, etc.) (Mel’čuk, 1996). When applied as a function to the semantic head of the expression, such an LF provides the second element.[4] There is a specific (simple or complex) LF for each recurrent co-occurrence pattern in language. LFs are shown to be language- and domain-independent. But note that LF instances are by nature language- and even domain-specific. Consider, for illustration, the entry for CONCENTRATION:

```
concentration {
    // Grammatical characteristics:
    dpos = N // deep part of speech is N(oun)
    spos = common_noun // surface part of speech is common noun
    // Government pattern (subcategorization):
    gp = {
        // Sem-DSynt valency projection (1⇒I, 2⇒II):
        1 = I // first semantic actant is first deep-syntactic actant
        2 = II // second semantic actant is second deep-syntactic actant
        // First syntactic actant can be realized as "ozone concentration":
        I = {
              dpos=N // actant is a noun
              rel=compound // linked with compound relation
              det=no // takes no determiner
        }
        // First syntactic actant can be realized as "concentration of ozone":
        I = {
          dpos=N // actant is a noun
          rel=noun_completive // linked with noun_completive relation
          prep=of // takes preposition "of"
          det=no // takes no determiner
        }
        // Second syntactic actant can be realized as "concentration of 180 μg/m3":
        II = {
            dpos=Num // actant is a number
            rel=noun_completive // linked with noun_completive relation
```

[4] An LF may provide as second element several alternative LUs; cf. *g*ive|*d*eliver|*m*ake| *a speech*. LFs are thus *maps* rather than functions. However, for convenience, LFs are usually referred to as functions in the literature.

```
            prep=of // takes preposition "of"
        }
    }
    // Lexical functions:
    Magn = high
    AntiMagn = low
    Adv1 = in // "(we found) ozone in a concentration (of 180 μg/m3)"
    Func2 = be // "the concentration (of ozone) is 180 μg/m3"
    Oper1 = have // "ozone has a concentration (of 180 μg/m3)"
    IncepFunc2 = reach // "the concentration (of ozone) reached 180 μg/m3"
    IncepOper1 = reach // "ozone will reach a concentration (of 180 μg/m3)"
}
```

We use two levels of granularity for the part of speech, referred to as *deep* and *surface* part of speech (resp. *dpos* and *spos*). This allows for quick reference to a whole family of parts of speech in grammar rules (for example, "N" refers to any proper noun, common noun, or pronoun). All specific grammatical characteristics of an LU would be described here as a feature-value pair (for example, its gender or its ability to take or not plural, definiteness, a certain tense, etc.).

The sub-categorization must contain the projection of the semantic to the syntactic valency and all possible ways of syntactically connecting the LU with its dependents. Governed prepositions must be indicated here, as well as case assignment if it exists in the language being described. One can optionally restrict the part of speech of the dependents (for instance, the first actant of CONCENTRATION must be a noun, while its second must be a number).

As mentioned above, lexical functions are an efficient way of referring to recurrent semantic and syntactic patterns of restricted lexical co-occurrence. In the example above, $Magn$ points to an LU which is a syntactic modifier and has a meaning of intensification ($AntiMagn$ is its antonym). The function $Func_2$ refers to a semantically emptied verb that takes the keyword (CONCENTRATION) as its subject and the keyword's second semantic actant as its object. $IncepOper_1$ points to a verb meaning roughly 'start' which takes the keyword as its object and the first actant of the keyword as its subject. The more information on restricted lexical co-occurrence the lexical dictionary contains, the more natural and idiomatic the generated text will feel.

MATE lets the user define as many dictionaries as necessary. We have presented the three main ones we have in our resources, but there can be more. For instance, it is possible to use a full-form dictionary instead of a proper morphological model, or a hybrid model as we implemented in our system. We also had a pseudo-dictionary for each language where we stored information such as the name of the language, the branch/family it belongs to, its being a "pro-drop language" or not, etc. This information forces or blocks the application of specific rules. For example, marking a language as "pro-drop" blocks the rules of SSynt $\Rightarrow$ DMorph that realize pronominal subjects.[5]

[5]We still need the pronoun in the surface syntactic structure in order to perform agreement, which

3.2 Grammar Encoding in MATE

A grammar $\mathfrak{G}^i_{i+1}$ consists of a set of minimal grammar rules of the following general format (see (Bohnet, 2006, 39ff) for details):

leftside (ls): $< g_i >$
rightside (rs): $< g_{i+1} >$
rightcontext (rc): $< g'_{i+1} >$
conditions (cd): <Boolean expr. over $\mathfrak{D}_{conc} \cup \mathfrak{D}_{sem} \cup \mathfrak{D}_{lex} \cup \mathfrak{S}_i \cup \mathfrak{S}_{i+1} >$
correspondences (cr): $\{n_{i_j} \Leftrightarrow n_{i+1_k}\}$

with g_i being a graph defined over the node and arc alphabets of $\mathfrak{S}_i$, g_{i+1} and g'_{i+1} being defined over the node and arc alphabets of $\mathfrak{S}_{i+1}$; $\mathfrak{D}_{conc}$, $\mathfrak{D}_{sem}$, $\mathfrak{D}_{lex}$ being the conceptual, semantic and lexical dictionaries; and $n_{i_j} \in g_i$, $n_{i+1_k} \in g_{i+1}$. The application of a rule consists in the identification of an isomorphic image of g_i in a given source structure S_{i_j} and subsequent introduction of an isomorphic image of g_{i+1} in the target structure S_{i+1_k} which is under construction. The statement '$n_{i_j} \Leftrightarrow n_{i+1_k}$' establishes a link between corresponding nodes in S_{i_j} and S_{i+1_k} in order to ensure that (i)information can be propagated from node to node across strata, (ii) the isolated fragments of the target structure as introduced by the individual rules can be unified to a connected well-formed structure. A rule is applicable if the specified conditions are fulfilled. As indicated, conditions may be defined over all dictionaries and both strata.[6] The rules in $\mathfrak{G}^i_{i+1}$ are minimal in the sense that the left-hand side of each rule is maximally elementary from the linguistic perspective: its g_i consists either of an elementary meaningful graph defined over the alphabets of $\mathfrak{S}_i$ or a graph that is transduced to an elementary meaningful graph defined over the alphabets of $\mathfrak{S}_{i+1}$. As a rule, an elementary meaningful graph consists either of a single node (a name) or a single arc (a linguistic relation)—although sometimes bigger structures are required.

In the remainder of this section, we illustrate the system with sample rules for the first four types of transduction involved in MTT-based generation.[7]

Rule 1 (Sample Con ⇒ Sem rule)

```
ls:  ?Xcon{PTIM->?T{con="tomorrow"}}
rc:  ?Xsem{tense=FUT}
cr:  ?Xcon ⇔ ?Xsem
```

Rule 1 maps the conceptual time relation between the concept denoted by the variable '?Xcon' and the "universal"[8] concept TOMORROW onto the tense feature

takes place in the SSynt ⇒ DMorph transduction.

[6] As a matter of fact, the conditions may also draw on the context, the discourse structure, the user model, etc. However, for simplicity's sake, we neglect this issue here.

[7] The rules of the DMorph ⇒ SMorph and SMorph ⇒ Text transductions are less interesting since they simply spell out morphological features of the words and pass the strings to an external morphological model.

[8] It is not absolutely true that all concepts are universal; some could be said "culture-specific". For instance, periods of the day vary considerably from one culture to another. In Spain, for example,

"FUT" of the semanteme denoted by the variable '?Xsem'. Note that '?Xsem' is specified in the right context slot—which means that the corresponding semanteme is assumed to have been already introduced into the target structure by another rule.

Rule 2 (Sample Sem ⇒ DSynt rule)

```
ls:  ?Xsem{?r->?Ysem}
rs:  ?Xds{I->?Yds}
rc:  ?Xds
cr:  ?Xsem ⇔ ?Xds
     ?Ysem ⇔ ?Yds
cd:  lexicon::(?Xds.lex).(gp).(?r)=I
```

Rule 2 maps any semantic relation (denoted by the variable '?r') of the semanteme denoted by '?Xsem' onto the first deep syntactic actant of the corresponding LU (denoted by '?Xds'). The node '?Xds' being also in the right context slot, must be already present in the target structure. This rule has a condition that accesses a dictionary called "lexicon" (which is the lexical dictionary introduced in Section 3.1). It searches for the entry that corresponds to the lexicalisation on the node '?Xds' and browses its attributes to verify that the projection of the semantic to the syntactic valency of the LU is such that the semantic relation '?r' is mapped to the deep-syntactic relation 'I'. For instance, this rule would apply to the first semantic argument of 'concentration' (cf. the sub-categorization for CONCENTRATION in Section 3.1). This rule can be further refined to handle any deep-syntactic actantial relation and to retrieve more information from the dictionary, such as grammatical features imposed on the actant by its governor (part of speech, mood, definiteness, etc.). For the sake of clarity, we shall consider only this simplified version.

Rule 3 (Sample DSynt ⇒ SSynt rule)

```
ls:  ?Xds{dpos=V; finiteness=FIN; mood=IND; tense=FUT}
rs:  ?Yss{slex=will
          dpos=lexicon::(will).dpos
          spos=lexicon::(will).spos
          tense=PRES
          finiteness=?Xds.finiteness; mood=?Xds.mood
          aux_completive->?Xss{finiteness=INF}
rc:  ?Xss
cr:  ?Xds ⇔ ?Yss
     ?Xds ⇔ ?Xss
cd:  language::(id).(iso)=ENG
```

Rule 3 introduces for an English verbal LU (referred to by '?Xds') that carries in the DSyntS the grammemes FIN, IND, and FUT the auxiliary WILL. WILL

the afternoon does not start before 3PM, while in Germany it starts as early as 12:00. We leave this problem aside as it is beyond the scope of this paper.

inherits from ?Xds the grammemes of finiteness and mood, but not of tense—which is for WILL PRES(ent) since the auxiliary itself has the present form (though it does express a future tense). Note that in this case, one deep-syntactic node corresponds to two surface-syntactic nodes.

Rule 4 (Sample SSynt ⇒ Top rule)

```
ls:  ?Xss{dpos=V
          subj-> ?Yss
          ?r-> ?Zss}
rc:  ?Ytp{b-> rc:?Ztp}
cr:  ?Ytp ⇔ ?Yss
     ?Ztp ⇔ ?Zss
cd:  not ?r=circumstantial
```

Rule 4 defines the relative ordering between the subject ('?Yss') of a verbal lexeme ('?Xss') and any other dependent of the verb ('?Zss'): the subject goes before. The circumstantials (roughly speaking, the adjuncts) are excluded since they may come before the subject.

4 Principles of Grammatical Resource Development

The sample rules cited above for illustration already give a hint that writing and maintaining comprehensive MTT-based generation grammar modules is a complex and very costly task—in particular, if the generation is to be multilingual.

To achieve the maximal efficiency possible, we adopt the following guidelines when organizing the grammatical resources:

(a) extracting recurrent core rule patterns across languages and factorizing them out into a "meta-grammar",

(b) modularizing language-specific rules,

(c) shifting the bulk of the grammarian's work to the lexicon,

(d) generalizing recurrent lexical patterns and introducing an inheritance mechanism.

Let us discuss the application of each of these guidelines in practice.

4.1 Sharing Core Grammar Components Across Languages

When developing grammatical resources for several languages in parallel, one quickly finds that many of the rules are the same in more than one language—to the point that some are identical for all languages under consideration. For instance, Rule 2 mentioned in Section 3.2 would apply no matter whether the language is

Catalan, English, French, Polish, Portuguese or Spanish. It makes no reference to any specific LU, nor does it refer to any language-specific relation (semantic and deep-syntactic relations being universal by definition). In that sense, it is a "universal" rule in our system. However, we do not claim that our resources contain even one single rule that could be applied in any possible language. Although Rule 2 seems a good candidate to universality (since it merely activates lexical information), consider for instance Rule 4 in Section 3.2. This rule also applies to all the languages we considered in our application (even supposedly "free-order" Polish looked better when the subject came first for the texts we had to generate). However, it is clear that it cannot apply to all known languages.[9]

In contrast with these two generic rules, Rule 3 given in Section 3.2 is only valid for English. It refers to a specific lexeme (WILL) and it even explicitly requires that the ISO identification code of the language being currently processed be "ENG" (see the note on the pseudo-dictionary for languages in Section 3.1).

Between these two extremes, it is possible to have rules that apply to a family (or any arbitrary set) of languages. For example, consider noun-determiner agreement. It does not exist in English nor in Polish. However, it is functionally the same in all Romance languages under consideration (the determiner agrees in gender and number with its governing noun). It is therefore possible to have only one rule for all those languages (cf. Rule 5).

Rule 5 (An SSynt ⇒ DMorph agreement rule for Romance languages)

```
ls:  ?Xss{dpos=N
          det->?Yss}
rc:  ?Ytp{gender=?Xss.gender
          number=?Xss.number}
cr:  ?Xtp ⇔ ?Xss
     ?Ytp ⇔ ?Yss
cd:  language::(id).(family)=romance
```

The general principle is to minimize the number of language-specific rules (such as Rule 3) and maximize the number of generic rules. The degree of generalization one can achieve for a module depends on the language and the strata involved. Languages that have agreement or a lot of lexical markers for grammatical meanings (articles, auxiliaries, etc.) require more language-specific rules. Table 1 shows, for each module in our system, the number of generic and language-specific rules we have and the percentage (in parentheses) of language-specific rules for each language within each module. The last row shows the percentage of language-specific rules for the individual languages over all modules.

As one can observe from Figure 2, the deeper the strata, the more generic a module tends to be. The Con ⇒ Sem module is entirely language-independent, as it relies on a more or less ad-hoc dictionary where a lot of information is hard-coded. It is however highly domain-specific. We do not consider it as part of the

[9] Cf., e.g., the word order in relative clauses in German.

Table 1: Number of generic and specific rules per module and language

Module	**Core**	CT	EN	ES	FR	PL	PT
Con ⇒ Sem	50	0	0	0	0	0	0
Sem ⇒ DSynt	59	8 (12%)	8 (12%)	7 (11%)	7 (11%)	11 (16%)	6 (9%)
DSynt ⇒ SSynt	64	13 (17%)	16 (20%)	11 (15%)	16 (20%)	7 (10%)	12 (16%)
SSynt ⇒ DMorph	70	13 (16%)	3 (4%)	12 (15%)	19 (21%)	8 (10%)	14 (17%)
DMorph ⇒ SMorph	7	8 (53%)	5 (42%)	6 (46%)	10 (59%)	10 (59%)	6 (46%)
SMorph ⇒ Text	12	1 (8%)	1 (8%)	1 (8%)	1 (8%)	1 (8%)	1 (8%)
Language-specific rules (%)		14%	11%	12%	17%	12%	13%

linguistic model as such, since it has to be rewritten for each application,[10] while the other modules are intended to be as domain-independent as possible.

The Sem ⇒ DSynt module has, on average, 12% of language-specific rules (with figures ranging from 9% for Portuguese to 16% for Polish). Language specific rules at this level are essentially for handling deep anaphora and some idiomatic expressions that cannot be captured by standard lexical functions, such as *in the afternoon* in a sentence like *In the afternoon, the ozone concentration was <will be> high*, which can be used only if the afternoon in question is already in the past or has not come yet (but not if it is present). The rest of the rules are generic and handle deep lexicalisation, syntactic tree building, support verbs described via standard lexical functions, quantification, etc. For example, Rule 2 in Section 3.2 activates the projection of the semantic to the syntactic valency found in the dictionary for any LU of any language.

The DSynt ⇒ SSynt module has an average of a little more than 16% language-specific rules. This ratio varies considerably from one language to another (from 10% for Polish to 20% for English and French). This is because auxiliaries, articles and all other grammatical words are handled at this level. Thus, languages with more lexical markers for grammatical meanings will require more specific rules in this module than languages that tend to express these meanings morphologically. For the treatment of verbal tense and aspect, we have a separate rule for each possible auxiliary combination (*will do*, *will be doing*, *will have done*, *will have been doing*, etc.). It would certainly have been possible to write only one rule for each auxiliary, with conditions handling the correct composition when more than one auxiliary is used. As a matter of fact, this would have better followed our general guidelines for grammar development (we tend to generalize the rules as

[10] We are investigating ways of automating this task.

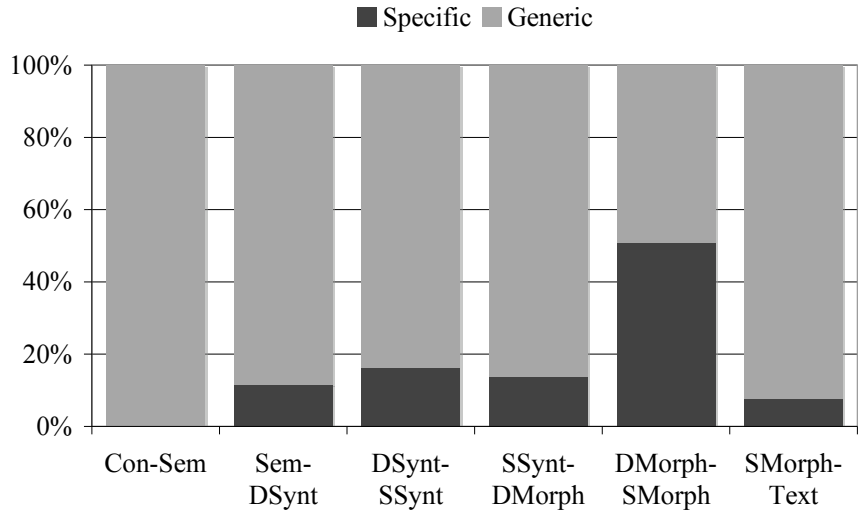

Figure 2: Average generic / language-specific rule ratio by module

much as we can, in order to keep a high maintainability of the resources). Doing so would have reduced the number of rules necessary for the auxiliaries from 12 to 4 for English. However, it would also have made those four rules significantly more complex. We preferred keeping a higher number of simpler rules, so that grammarians with less experience in formal linguistics would easily understand, maintain and port them to other languages.

The SSynt ⇒ DMorph rules model two main phenomena: word-order and agreement. Both phenomena vary greatly from one language to another. It is therefore a little surprising that we have less than 14% language-specific rules on average here. We believe that this is something of a statistical anomaly in that most of the time, the best word-order for the texts we had to generate was more or less the same for all languages considered in this application, including Polish. Obviously, if we had to generate other kinds of texts or in other languages, we would certainly see the number of language-specific rules go up for this module. However, the good news is that word-order rules are among the simplest, so writing and maintaining them is easy.

The DMorph ⇒ SMorph module shows the highest ratio of language-specific rules: almost 51% on average (ranging from about 42% for English to nearly 59% for French and Polish). However, this module contains very few rules (only 12 to 17 rules depending on the language, of which 7 are generic). It prepares the strings that will be passed to the morphological module (or the full-form dictionary), with all grammatical features in the correct order (so that an English finite verb, for instance, would look like “reach<V><IND><PRES><SG><3>”). Basically, most language-specific rules of this module only recopy the attributes found on the nodes at previous levels as explicit codes in the chain that labels the node. It is a purely

technical process that has little linguistic relevance, but the number of language-specific rules still has a strong correlation with the nature of the language: the more complex the morphology of the language is, the higher the number of language specific rules will be. It is also in this module that operations such as elision (Fr. *le homme → l'homme*), contraction (Eng. *does not → doesn't*) and epenthesis (Pl. *w wjezdzie → we wjezdzie*) are computed.

The number of language-specific rules in the SMorph ⇒ Text module[11] is quite low (8% for all languages). However, this figure does not reflect any interesting linguistic fact. This module only manages the final realization of wordforms; it handles spelling out, capitalization, and so on. The number of rules in this module is very low (13 only for any language, of which 12 are generic). The only rule that is language-specific simply calls the appropriate two-level morphology model or full-form dictionary for the language in question, passing on the string that was built by the previous module.

Given that in each module we need some language-specific rules, the rules are sensitive to the language that is being processed. While most can always apply, some are marked in order to apply (or not to apply) for a given language (or set of languages). In fact, what we have is a set of rules from which the grammar of a specific language is a subset. From the point of view of the developer, it can be seen as a pool of shared rules to which one can "subscribe" for the language he wants to describe.

4.2 Rule packaging

The rules inside each module are further organized into packages. A package is a set of rules that work together, or on the contrary, are in competition; it handles one specific linguistic phenomenon. For example, in the DSynt ⇒ SSynt module, there are separate packages for idioms, coordination, auxiliaries, etc. Formally, a package is defined by an abstract rule from which other rules depend. An abstract rule is always empty, but it may have conditions associated to it, which are inherited by all the rules that depend on it. A package can be composed of sub-packages. It is notably the case of the language packages. Language-specific rules are grouped into a separate package for each language, which consists of a number of sub-packages for various language-specific phenomena (see Section 4.1 for a list of such phenomena by module).

In each module, there is a so-called "core" package that contains all essential rules that are needed for processing any input structure. For instance, the SSynt ⇒ DMorph module's core rules handle lexicalisation, actantial relations and the ATTR-relation (for modifiers), without which nothing can be done. Also in this module are packages for lexical co-occurrence (in particular support verbs), quantification, circumstantials, voice, and language-specific packages (mainly for deep anaphors). Phenomena that span over more than one module, such as coordination,

[11] Recall that we use an additional transduction from the SMorphS to text not foreseen in MTT.

have a corresponding package in each module involved.

With this design it is possible to assign different packages to different developers who are specialists of a specific domain and who need not worry about the problems outside of their sandbox. It is also easier to modify the grammar to meet specific needs by choosing the desired modules or by adding in new ones.

So far, we have limited the package-based design to grammars only. Eventually, we will also adopt the same architecture for the dictionaries. The lexical core of each language should constitute the main package, while additional packages could be developed by qualified lexicographers for specific domains (air quality, traffic information, healthcare, etc.).

4.3 Rich Hierarchical Dictionaries

As is customary in many modern grammar theories (among others, HPSG, SFG, Word Grammar, etc.) and their implementations, we use inheritance in the lexical resources, factorizing all possible lexical information into abstract entries from which the LUs depend. Consider, for illustration, a fragment of the verbal hierarchy $predicate \rightarrow verb \rightarrow direct\ transitive\ verb$.

The $predicate$ node provides the default projection of the semantic valency to the syntactic valency of a predicative unit. We assume that a predicate possesses at most six actants, with the ith semantic actant (denoted by an Arabic numeral) usually corresponding to the ith syntactic actant (denoted by a Roman numeral):

```
"predicate" {
   gp={ 1=I; 2=II; 3=III; 4=IV; 5=V; 6=VI } }
```

A verbal lexeme is a predicate (i.e., inherits, if not overridden, all features defined for the $predicate$ unit). Furthermore, its surface and deep part of speech are respectively 'V' and 'verb' and, by default, its first syntactic actant is realized as a grammatical subject, usually a noun (this can of course be overwritten for any given verb):

```
"verb" :  "predicate" {
   dpos=V; spos=verb
   gp={ I= {dpos=N; rel=subj} }
}
```

Note that such abstract entries are not necessarily universal. For each language, we keep a separate hierarchy since the parts of speech and the morpho-syntactic behavior of their members can vary cross-linguistically. For example, Polish verbs usually assign the nominative case to their subject, unless otherwise specified, so this information would be added to the abstract $verb$ entry for Polish. One could prefer having a $polish\_verb$ entry that would inherit from the $verb$ entry above (or even from a more generic one that would not specify the nature of the subject, for instance) and refine it. However, it is not possible in the current version of the

system to have an entry shared by several languages. All that can be done is to copy the $verb$ entry into the respective dictionary of each language. Since it was not possible to have the information written once for all languages, and since the differences between the parts of speech of the languages we had to deal with were not great, we did not seek a more refined hierarchy.

English direct transitive verbs inherit from the $verb$ class. Furthermore, they realize their second syntactic actant as a direct object, and it is by default a noun:

```
"verb_dt" :  "verb" {
   gp={ II= {dpos=N; rel=dobj} }
}
```

Now, adding a direct transitive verb to the lexicon is just a matter of expressing its membership in the $verb\_dt$ class. Consider, for illustration, the entry for the verb EXCEED below, where we have added information on its lexical co-occurrence:

```
exceed :  "verb_dt" {
   Magn = "by far"
   AntiMagn = "a little"
}
```

All information about the projection of the semantic to the syntactic valency, part of speech and surface realization of the actants has been inherited. Of course this information can be overridden, simply by overwriting it. For example, the verb EXPECT has two possible sub-categorization patterns, none of which corresponds to the default pattern for verbs:

```
expect :  "verb" {
   gp={ II={dpos=V; finiteness=FIN; mood=IND; prep=that} }
   gp={ II={dpos=V; finiteness=INF; prep=to; rel=iobj}
        raise={ II={rel=dobj} } }
}
```

The first pattern corresponds to *We* [=I] *expect that the ozone concentration will increase* [=II]. The second pattern points to a subject-raising construction where the subject of actant II is raised to become the direct object of EXPECT, downgrading actant II to an indirect object position, as in *We* [=I] *expect the ozone concentration* [=raised subject] *to increase* [=II].

5 Benefits of the Proposed Grammar Design

The principles outlined above and followed in our work ensure that

(i) no parts of resources are repeated,

(ii) the resources are linguistically sound, and

(iii) the acquisition and maintenance (i.e., evaluation, correction and extension) of the resources can be carried out easily by grammarians without an extensive training in the linguistic theory underlying the generator.

To be underlined in particular are the extensibility to new languages and the extension towards the coverage of new linguistic constructions. Thus, this design allows for quick addition of new languages to the generator. Few changes need to be made to the grammar rules, since most of the language-dependent information is in the dictionaries. Rules that take care of articles, auxiliaries and other lexical markers of grammatical meanings, as well as agreement and word-order rules do need to be modified, but they are usually very simple. Hence, the task of adding a new language basically boils down to describing the LUs of the language in question.

Similarly, adapting the generator to a new domain is even more simple. Many of the existing dictionary entries can be reused, and one only needs to describe the new lexemes found in the vocabulary of the new domain. As a matter of fact, while the grammar described here was first used for a project in the domain of air quality, we have successfully reused the surface modules (from SSynt $\Rightarrow$ DMorph) in another project for a totally different domain (patents on optical recording technology) with very little modifications to the rules.

A benefit that is not to be underestimated for the design described here is the ease of development that follows from it in terms of work organization. Broad coverage grammars, especially in a multilingual context, require a relatively large team. It is then unrealistic to hope for a homogeneously qualified team who can work on any aspect of the problem, in particular within a lesser-taught framework such as MTT. Fine-grained modularity allows for efficient task separation. The number of language-specific grammar rules being kept as low as possible, most of the work for adding a new language lies in writing the dictionaries for it.

One could argue that all we have done was just to shift the workload from the grammar to the dictionary. It is true to some extent, but the formalism used for grammar rules is much more complex than the one used for dictionaries. We have shown here simplified rules, but a serious grammar cannot consist only of such simple rules. They can get rather complex, enough to scare away a potential contributor who is not necessarily very comfortable with formal languages. Dictionaries, however, are written in a very simple formalism that can be mastered quickly. Indeed, our experience showed that it was much easier for people to learn how to write dictionaries than how to write grammar rules. Hence, by shifting the workload to the dictionary, we simplify the process of describing new languages as resources become more easily maintainable and extensible. By adopting this architecture, we are able to have a (more or less) permanent core team that has enough experience with the grammar formalism to work on the generic rules, and short-term collaborators who can join for a specific project to develop resources for a new language without having to learn in detail the formalism used for the codification of grammar rules.

6 Evaluation

In accordance with the evaluation principles in software engineering, we use a twofold evaluation procedure: what we may call *micro-testing* and *macro-testing*. We also built a tool to verify the consistency of our dictionaries.

6.1 Micro-testing

Micro-testing refers to the evaluation of single rules. For each rule, a set of test structures has been set up. These structures must be as simple as possible, in order to avoid noise, but still designed to cover all the phenomena the rule has to be able to cope with. In most cases, a single rule cannot be tested in isolation; its application depends on the application of some core rules. For instance, it is not possible to test only the construction of a given syntactic relation without also applying the rules that create the nodes linked by the relation. Therefore, it is necessary for evaluation to keep track of rule dependencies. Hence, not only do we associate a set of test structures with each rule, but we also associate each test structure with the set of rules it activates. When a rule is modified, all executed test routines that involved it are reset and run again.[12] Micro-testing thus verifies elementary components separately.

By the very nature of micro-testing, it is difficult to have language-independent test structures. For example, even if one wants to test a generic DSynt $\Rightarrow$ SSynt rule, the input test structure will have to be a DSynt structure, which by definition contains LUs of a specific language. Therefore, only the rules of the Con $\Rightarrow$ Sem module can be micro-tested with language-independent test structures (since our conceptual structures are the same for all languages). However, it is often safe to assume that generic rules tested in one language will work just as well in other languages, though of course prudence must be used. Figure 3 shows a DSynt structure that we used for testing the rules that handle subject-raising verbs. Though this structure uses information from the English dictionary, it tests rules that are generic.

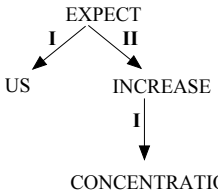

Figure 3: A sample DSynt structure for testing subject-raising

[12] Of course, this process can be automated if we have, for all test structures, the expected result structures (be they manually created from scratch, or result from previous test runs that have been validated by a human).

Phenomena that are described by language-specific rules obviously require language-specific structures for micro-testing. For instance, rules that handle English auxiliaries were tested with a set of nearly identical structures containing only a verb with a subject and an object, where the sole difference was the tense and aspectual information. One structure was created for each possible combination of tense and aspect.

6.2 Macro-testing

Macro-testing refers to a more global evaluation procedure. Its aim is to assess the coverage of the linguistic resources for a more or less specific purpose. The structures used for this task are designed to cover the largest possible range of situations the generation system must be able to handle. The goal here is not to test specific rules, but to make sure that the system can handle the expected input we are going to provide it with. Macro-testing is best applied after micro-testing, as it verifies the interaction between the various components of the grammar. For example, Figure 4 shows a structure that was used for macro-testing.[13] The system is required to produce all expected ways to express this meaning:

> *Between 8 AM and 11 AM, the concentration of sulfur dioxide remained stable at 3.*
> *Between 8 AM and 11 AM, the sulfur dioxide concentration was stable at 3.*
> ...

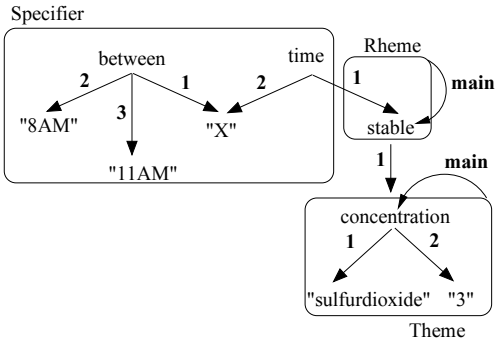

Figure 4: A sample semantic structure for macro-testing

6.3 Dictionary consistency testing

In addition to micro- and macro-testing, it was necessary to make sure that all concepts that might appear in the input structures could be expressed in any lan-

[13] We show here an English semantic structure, though the real input structure is a conceptual structure. The corresponding conceptual structure takes too much space and would be difficult to read, so for the sake of clarity we show only this semantic representation.

guage. Concepts are mapped by the conceptual dictionaries to language-specific semantemes, which in turn are mapped to LUs. These LUs point to prepositions in their sub-categorization patterns, and to other LUs through lexical functions. All these links form a complex network where errors are hard to spot for a human, so we created a small MATE grammar that consisted essentially of simplified lexicalisation rules. This grammar takes as input a list of structures containing one concept each (one structure for every concept expected in the input of the system) and produces structures representing the lexical links encoded by the dictionaries. Then, a set of consistency-check rules is applied to make sure that there is no pointer to non-existent entries, and that each entry contains all the necessary information (for example, that French nouns have a gender, that every LU has a part of speech, that syntactic relation names are specified in the sub-categorization patterns, etc.). If an error is found, an appropriate message is added to the output in the form of extra nodes and relations, as illustrated in Figure 5, where IN is marked as missing in the lexical dictionary. MATE's graphs being encoded as text files, it is easy to scan the output structures for error messages (or have a script do it).

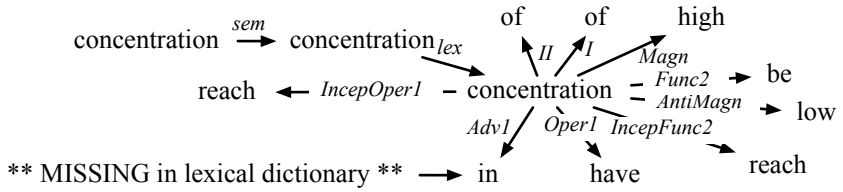

Figure 5: A sample output structure from the lexical consistency check grammar

7 Conclusion

We presented an efficient organization of grammatical resources in an MTT multilingual generation system. This organization follows the principles of sharing, modularization, and inheritance and adopts a strongly lexicalist perspective on the grammatical resources. The implementation of resources for six languages that belong to three different families and their practical use have proven that these principles are valid and allow for the development of large scale grammars.

As part of future work, we plan to extend the resources with respect to both the coverage of linguistic constructions and further languages.

References

Alshawi, Hiyan. 1992. *The Core Language Engine*. Cambridge, MA: The MIT Press.

Avgustinova, Tania and Uszkoreit, Hans. 2000. An ontology of systemic relations for a shared grammar of Slavic. In *Proceedings of the 18th International Conference on Computational Linguistics, COLING*, pages 28–34.

Bateman, John, Kruijff-Korbayova, Ivana and Kruijff, Geert-Jan. 2005. Multilingual resource sharing across both related an unrelated languages: An implemented, open-source framework for practical natural language generation. *Journal for Research on Language and Computation* 3(2), 191–219.

Bender, Emily M., Flickinger, Dan and Oepen, Stephan. 2002. The Grammar Matrix: An Open-Source Starter-Kit for the Rapid Development of Cross-Linguistically Consistent Broad-Coverage Precision Grammars. In John Carroll, Nelleke Oostdijk and Richard Sutcliffe (eds.), *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics, COLING*, pages 8–14, Taipei, Taiwan.

Bohnet, Bernd. 2006. *Textgenerierung durch Transduktion linguistischer Strukturen*. Berlin: Akademische Verlagsgesellschaft, DISKI Series.

Bohnet, Bernd, Langjahr, Andreas and Wanner, Leo. 2000. A Development Environment for MTT-Based Sentence Generators. In *Proceedings of the XVI SEPLN Conference*, Vigo, Spain.

Gerdes, Kim and Kahane, Sylvain. 2007. Phrasing It Differently. In L. Wanner (ed.), *Selected Lexical and Grammatical Issues in the Meaning-Text Theory. In honour of Igor Mel'čuk*, pages 297–335, Amsterdam: Benjamins Academic Publishers.

Kim, Roger, Dalrymple, Mary, Kaplan, Ronald M., Holloway King, Tracy, Masuichi, Hiroshi and Ohkuma, Tomoko. 2003. Multilingual Grammar Development via Grammar Porting. In *ESSLLI 2003 Workshop on Ideas and Strategies for Multilingual Grammar Development*.

Mel'čuk, Igor. 1988. *Dependency Syntax: Theory and Practice*. Albany, NY: SUNY Press.

Mel'čuk, Igor. 1996. Lexical Functions: A Tool for the Description of Lexical Relations in a Lexicon. In L. Wanner (ed.), *Lexical Functions in Lexicography and Natural Language Processing*, pages 37–102, Amsterdam: Benjamins Academic Publishers.

Santaholma, Marianne. 2007. Grammar Sharing Techniques for Rule-Based Multilingual NLP Systems. In *Proceedings of NODALIDA 2007*, pages 253–260.

Sowa, John. 2000. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Pacific Grove: Brooks Cole Publishing Company.