The MetaGrammar: a cross-framework and cross-language test-suite generation tool

Alexandra Kinyon and Owen Rambow
University of Pennsylvania
kinyon@linc.cis.upenn.edu

Abstract

In this paper, we present a novel approach for building parallel syntactically annotated sentences for different frameworks and languages. These syntactically annotated sentences are automatically generated from a compact higher level of syntactic abstraction, **a MetaGrammar hierarchy**, which minimizes the need for human intervention in building annotated test-suites. With a single hierarchy, each sentence generated is automatically mapped to several annotations (traditional constituent structure, LFG F-structure, dependency structure and constraint grammar representation) thus allowing one to directly compare the coverage of grammars based on different frameworks. Furthermore, also with that same single compact hierarchy, the tool generates parallel annotated sentences for different languages, which may prove useful for Machine Translation evaluation.

1 Introduction

A large number of grammar formalisms exist for which expensive dedicated tools such as wide-coverage grammars, parsers etc. have been developed. All these grammar formalisms have the same goal: formalize the syntax of natural language so that it can be processed by a computer. However, they resort to very different means and to very different formats of representations to achieve this goal. Broadly speaking, some formalisms rely on the notion of Phrase Structure, others rely on the notion of syntactic function, others on the notion of dependency, and yet again others on the notion of "constraint" or on a hybrid type of syntactic information.[1]

As pointed out in (Sutcliffe et al., 1996), this variety of formats becomes a problem when one wants to directly compare grammars. One standard way to evaluate the coverage of a grammar consists in comparing the analysis of the grammar to a gold standard analysis i.e. hand-corrected syntactically annotated data. Traditionally, when talking about syntactically annotated data, one distinguishes treebanks, such as the Penn Treebank (Marcus et al., 1993), which consist of naturally occurring text and often contain complex syntactic phenomena and a large lexicon, from test-suites, such as TSNLP (Lehman and al, 1996). A test suite is a set of artificially built sentences, which are sorted by linguistic phenomena, and designed to contain a limited lexicon.[2] Evaluation approaches which rely on treebanks tend to be favored by the statistical parsing community because of their empirical flavor. However, rather than opposing treebanks to test-suites, it seems more fruitful to see both as complementary tools, each having its advantages and drawbacks for evaluation purpose. When evaluating the performance of a grammar against a treebank, one can obtain **quantitative** measures by computing metrics such as precision, recall, crossing brackets etc.[3] However, one can not obtain **qualitative** measures i.e. which syntactic phenomena are handled by the grammar. Since test-suite sentences are sorted by phenomena (e.g. *wh*-questioned object, passive with no agent, ...), when evaluating

[1]Sometimes, the dichotomy phrase-structure versus dependency structure is deemed language dependent because phrase structure representations are thought to be less adequate to formalize the syntax of so-called *free word order languages*, but even this is not to be taken for granted since there are, for instance, dependency grammars for English and phrase structure grammars for Japanese, such as the HPSG grammar of (Y.Mitsuishi et al., 1998).

[2]Another characteristic of test-suite is that they may contain ungrammatical sentences on purpose, but we do not address this issue here.

[3]For an overview of the various metrics used with their respective advantages and drawbacks, see e.g. (Carroll et al., 1998)

the performance of a grammar against a test-suite, one may find, in addition to the metrics above, which syntactic phenomena are badly handled (or not handled at all) by the grammar, thus giving a clear indication as to which grammar rules need to be added or improved. Moreover, when developing a grammar within a given framework, for a given domain and a given language, more often than not, an annotated treebank for this specific <framework,domain,language> triplet will not be available. For applications requiring parallel grammars for several languages, the problem is even more acute since virtually no treebank currently exists with parallel annotations for naturally occurring text in more than one language.[4]

However, both Treebanks and test-suites are costly to build: usually, once the data to be annotated is chosen (be it real world-data or artificial data), it is then parsed and hand-corrected. The hand-correction phase requires extensive human intervention. We propose to depart from this 2-phase scheme of annotation. Instead, we encode a compact higher level of syntactic abstraction, the MetaGrammar (MG), from which annotated data is generated. This allows us to make the development of test-suites faster by reducing the need for human intervention, and also by making it possible to generate, from a single MG hierarchy, parallel test-suites for different languages and frameworks.[5]

In the first part of this paper, we introduce the notion of MetaGrammar and explain how it can be used to classify and generate syntactically annotated sentences.

In the second part of this paper, we explain how the abstract level allows to "share" syntactic knowledge among different languages, and discuss how we have generated parallel annotated data for English and German.

In the third part of this paper, we explain how this approach has allowed us to generate in parallel the same data annotated for different syntactic frameworks. More specifically, we focus on phrase structure annotation, dependency annotation, LFG F-structure annotation and constraint grammar annotation.

Finally, in the last part of this paper, we discuss directions for future work, including the question of extending the approach to naturally occurring text in order to bridge the gap between treebanks and test-suites.

2 What is a MetaGrammar?

2.1 Theoretical Issues

The notion of MetaGrammar was originally presented in (Candito, 1996).[6] Her goal was to automatically generate a wide-coverage Tree Adjoining Grammar (TAG)[7] while minimizing human intervention in the grammar writing process, and thus easing grammar development and maintenance. The idea is to add a higher-level and compact layer of linguistic description, which imposes a general organization for syntactic information in a three-dimensional hierarchy :

- Dimension 1: initial subcategorization
- Dimension 2: valency alternations and redistribution of functions
- Dimension 3: surface realization of arguments.

Each terminal class in dimension 1 describes a possible initial subcategorization (i.e. transitive, ditransitive etc...). Each terminal class in dimension 2 describes a list of ordered redistributions of functions (e.g. it allows to add an argument for causatives, to erase one for passive with no agents ...). Each terminal class in dimension 3 represents the surface realization of a surface function (ex: declares if a direct-object is pronominalized, wh-extracted, etc.). Each class in the hierarchy is associated to the partial description of a tree (Rogers and Vijay-Shanker, 1994) which encodes *father*, *dominance*, *equality* and *precedence* relations between nodes. A well-formed tree is generated by inheriting from exactly one terminal class from dimension 1, one terminal class from dimension 2, and *n* terminal classes from dimension 3 (where *n* is the number of arguments of the elementary

[4]This may partly explain why industrial research project, such as Xerox XLE, still develop large hand-crafted grammars instead of inducing them from treebanks.

[5]Another advantage of our approach is that there is not need for a parser, and for disambiguation (manual or automatic) in order to build a test-suite.

[6]A similar metagrammar tool for TAGs may be found in (Xia, 2001)

[7]We do not present TAGs here, since it is orthogonal to our concern. Suffi ce it to say that the basic rules are phrase-structure trees.

tree being generated). For instance the elementary tree for *Par qui sera accompagnée Marie* (By whom will Mary be accompanied) is generated by inheriting from *transitive* in dimension 1, from *passive-no-agent* in dimension 2 and *subject-nominal-inverted* for its subject and *questioned-object* for its object in dimension 3. This compact representation allows one to generate a 5000 tree grammar from a hand-crafted hierarchy of a few dozens of nodes, esp. since nodes are explicitly defined only for **simple** syntactic phenomena.[8]

This particular MG tool was used to develop a wide-coverage TAG for French (Abeille et al., 1999) (5000 grammar rules), as well as a medium-size TAG for Italian (Candito, 1999). It is worth mentioning that in addition to proposing a compact representation of syntactic knowledge, (Candito, 1999) began exploring whether some components of the hierarchy could be re-used across similar languages (French and Italian). However, she developed two distinct hierarchies to generate grammars for these two languages and generated only grammars for the TAG framework [9]. We extend the use of the MetaGrammar, whose basic role is to generate trees, in order to generate annotated strings i.e. sentences in a test-suite. In order to achieve this, we simply add one dimension to the organization above: a **Lexicalization dimension** which allows us to specify lexical items i.e. the words which will appear in the annotated sentence. We also push further the cross-language and cross-framework potential of a MetaGrammar organization by generating annotated sentences for English and German from one single hierarchy and for different frameworks: traditional phrase structure, LFG F-structure, Dependency tree, constraint grammar annotation.

In addition to that, the syntactically annotated sentences we generate are classified by syntactic phenomena, thanks to the notion of *Hypertag*, which was introduced in (Kinyon, 2000). The main idea behind hypertags is to keep track, when trees are generated from a MG hierarchy, of their salient syntactic characteristics i.e. the terminal classes used for generating the tree[10]. For instance, the verb *give* in *A book was given to May* could be assigned the hypertag:

$$\begin{bmatrix} \text{Subcat: Ditransitive} & \\ \text{Valency alternations: Passive no Agent} & \\ \text{Argument Realization} & \begin{bmatrix} \text{Subject:} & \text{Canonical NP} \\ \text{Object:} & \text{Canonical NP} \end{bmatrix} \end{bmatrix}$$

Although we retain the linguistic insights presented in (Candito, 1996), that is the notions of multi-dimensions to model syntax, with one dimension for subcategorization, one dimension for valency alternation, one dimension for the realization of syntactic argument, and one dimension for the realization of lexical items, we use a different MetaGrammar tool which is less framework-dependent and supports the notion of hypertag.

2.2 Practical Issues

The MG compiler we use for generating test-suites is presented in (Gaiffe et al., 2002).[11] In the (Gaiffe et al., 2002) tool, each class in the MG hierarchy encodes:

- Its SuperClasse(s)
- A Hypertag which captures the salient linguistic characteristics of the class.
- What the class needs and provides
- A set a quasi-nodes
- Constraints between quasi-nodes (father, dominates, precedes, equals)
- Traditional feature equations for agreement.

The MG tool automatically crosses the nodes in the hierarchy, looking to create "balanced" classes, that is classes that do not need nor provide anything.[12] Then, for each balanced terminal classes, the structural constraints on quasi-nodes are unified, and if the unification succeeds, a pair <description,tree> is generated. Figure 1 shows a simple example of how one can generate the annotated sentences *The boy sees a duck* for phrase structure.

[8]Nodes for complex syntactic phenomena are generated by automatic crossings of nodes for simple phenomena

[9]For croos-language grammar development, one can also cite the LFG Parallel Grammar project (Butt et al., 2002), but to the best of our knowledge, this project does not explicitly explore rule-sharing across languages.

[10]The notion of hypertag was inspired by that of supertags presented in (Srinivas, 1997), which consists in assigning a TAG elementary tree to each lexical item in a sentence, hence enriching traditional POS tagging information. However, hypertags are framework-independent.

[11]This compiler is freely available on http://www.loria.fr/equipes/led/outils/mgc/mgc.html

[12]Another way to view this is by analogy to the notion of resource allocation graphs.

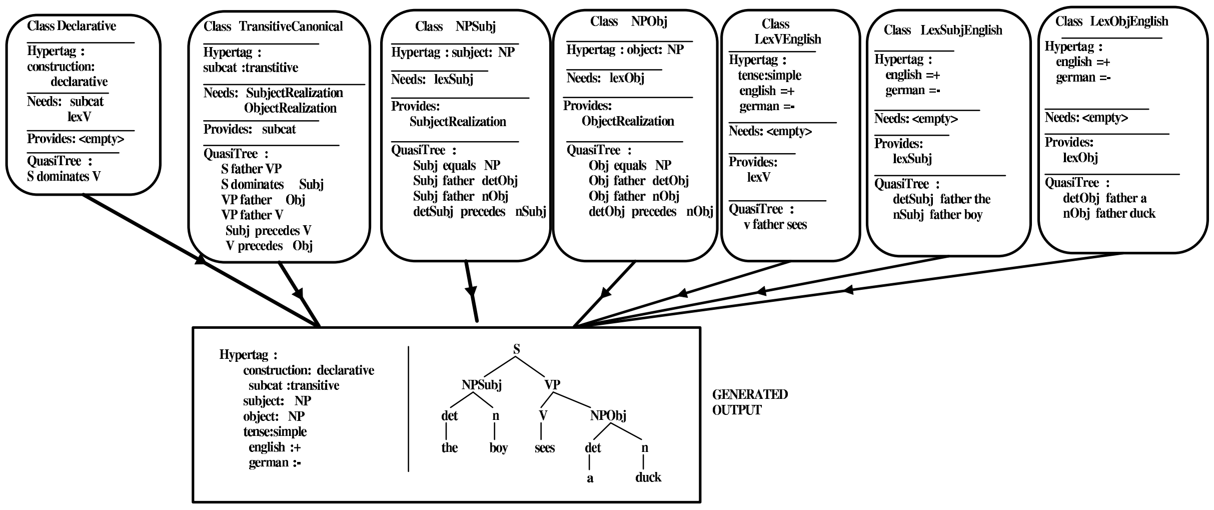
Class Declarative
Hypertag : construction: declarative
Needs: subcat lexV
Provides: <empty>
QuasiTree : S dominates V
Class TransitiveCanonical
Hypertag : subcat :transtitive
Needs: SubjectRealization ObjectRealization
Provides: subcat
QuasiTree : S father VP S dominates Subj VP father Obj VP father V Subj precedes V V precedes Obj
Class NPSubj
Hypertag : subject: NP
Needs: lexSubj
Provides: SubjectRealization
QuasiTree : Subj equals NP Subj father detObj Subj father nObj detSubj precedes nSubj
Class NPObj
Hypertag : object: NP
Needs: lexObj
Provides: ObjectRealization
QuasiTree : Obj equals NP Obj father detObj Obj father nObj detObj precedes nObj
Class LexVEnglish
Hypertag : tense:simple english =+ german =-
Needs: <empty>
Provides: lexV
QuasiTree : v father sees
Class LexSubjEnglish
Hypertag : english =+ german =-
Needs: <empty>
Provides: lexSubj
QuasiTree : detSubj father the nSubj father boy
Class LexObjEnglish
Hypertag : english =+ german =-
Needs: <empty>
Provides: lexObj
QuasiTree : detObj father a nObj father duck
Hypertag : construction: declarative subcat :transitive subject: NP object: NP tense:simple english :+ german :-
S
NPSubj
VP
det
n
V
NPObj
the
boy
sees
det
n
a
duck
GENERATED OUTPUT

Figure 1: Generation of sentence *The boy sees a duck*

Note that with the same hierarchy, but with slightly different dominance and precedence constraints on the hierarchy nodes, one obtains a sentence annotated for phrase-structure, but with different linguistic choices (e.g. an X-bar representation of the sentence).

A first advantage of the approach is that the syntactic phenomena covered are quite systematic and the annotation coherent: if sentences are generated for "transitive-passive-whExtractedByPhrase" (e.g. *By whom was the mouse eaten?*), and if the hierarchy includes ditransitive verbs, then the automatic crossing of phenomena ensures that sentences will be generated for "ditransitive-passive-whExtractedByPhrase" (i.e. By whom was Peter given a present).

A second advantage of the approach is to minimize the need for human intervention in the test-suite building process. Human intervention is needed to "organize" the linguistic knowledge i.e. encode the hierarchy, and then to verify that the output sentences are valid. There is no annotation phase after the generation. If the generation is not satisfactory, either because some sentences are ungrammatical, or some syntactic structures are deemed incorrect, then changes are made directly in the MG hierarchy and never in a post-generation phase. This ensures a homogeneity of the annotation which is not necessarily present with traditional hand-annotation approaches.

A third and essential advantage is that it is straightforward to have parallel annotations generated from a single hierarchy. We devote the next two sections to this topic.

3 Generating Cross-Language Annotated Data

Traditionally, phrase-structure grammars are deemed not well suited for languages such as German which exhibit a relative free-word order. The English sentence *today the boy sees a duck* has only 3 possible orders (with *today* at the beginning or at the end of the sentence, or, felicitous in some contexts, just before the verb). The same sentence in German, *heute sieht der Junge eine Ente*, has 6 possible word orders, since all the combinatorics between the constituents are allowed as long as the finite verb is in second position (the "V2" effect). Moreover, German also allows an expletive construction: *es sieht der Junge eine Ente heute*, thus doubling the number of possible word orders to 12. However, the MG tool allows us to generate all the possible word-orders, simply by leaving some precedence relations unspecified. In the appendix, we show how 18 English sentences sorted by syntactic phenomena correspond to 82 sentences in German because of word order variations. This total of 100 sentences covers for English and German declarative sentences, sentences with subordinate clauses with or without a complementizer, sen-

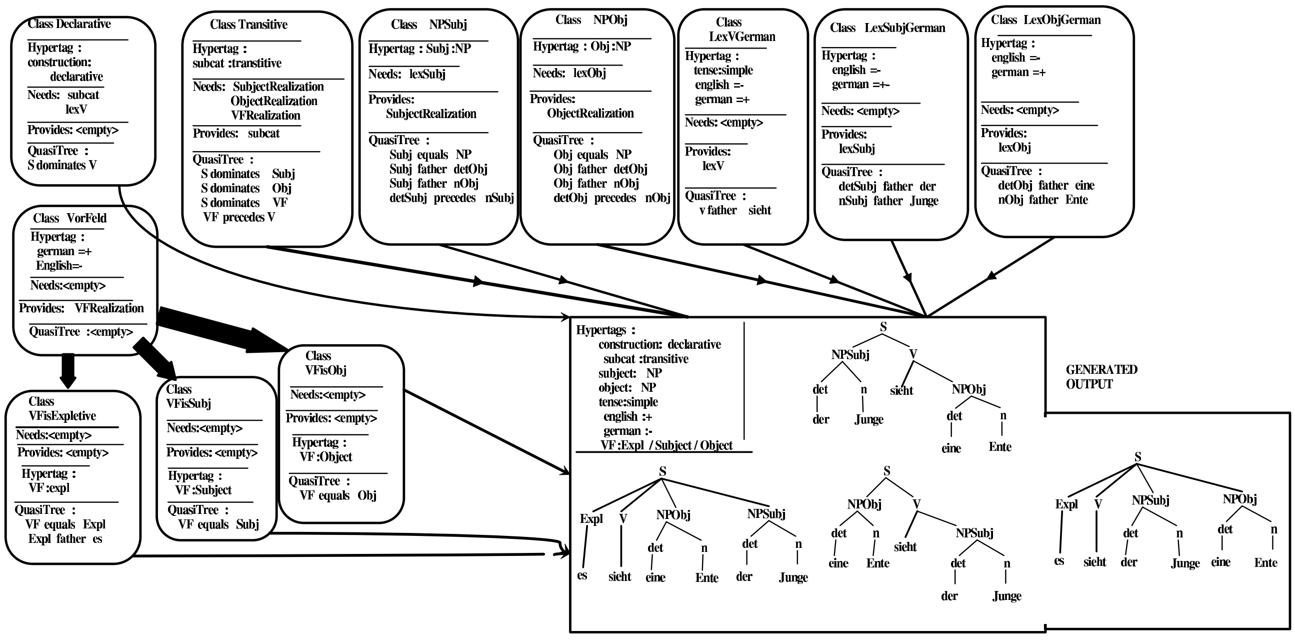
Class Declarative
Hypertag : construction: declarative
Needs: subcat lexV
Provides: <empty>
QuasiTree : S dominates V
Class Transitive
Hypertag : subcat :transtitive
Needs: SubjectRealization ObjectRealization VFRealization
Provides: subcat
QuasiTree : S dominates Subj S dominates Obj S dominates VF VF precedes V
Class NPSubj
Hypertag : Subj :NP
Needs: lexSubj
Provides: SubjectRealization
QuasiTree : Subj equals NP Subj father detObj Subj father nObj detSubj precedes nSubj
Class NPObj
Hypertag : Obj :NP
Needs: lexObj
Provides: ObjectRealization
QuasiTree : Obj equals NP Obj father detObj Obj father nObj detObj precedes nObj
Class LexVGerman
Hypertag : tense:simple english =- german =+
Needs: <empty>
Provides: lexV
QuasiTree : v father sieht
Class LexSubjGerman
Hypertag : english =- german =+-
Needs: <empty>
Provides: lexSubj
QuasiTree : detSubj father der nSubj father Junge
Class LexObjGerman
Hypertag : english =- german =+
Needs: <empty>
Provides: lexObj
QuasiTree : detObj father eine nObj father Ente
Class VorFeld
Hypertag : german =+ English=-
Needs:<empty>
Provides: VFRealization
QuasiTree :<empty>
Class VFisExpletive
Needs:<empty>
Provides: <empty>
Hypertag : VF :expl
QuasiTree : VF equals Expl Expl father es
Class VFisSubj
Needs:<empty>
Provides: <empty>
Hypertag : VF :Subject
QuasiTree : VF equals Subj
Class VFisObj
Needs:<empty>
Provides: <empty>
Hypertag : VF :Object
QuasiTree : VF equals Obj
Hypertags : construction: declarative subcat :transitive subject: NP object: NP tense:simple english :+ german :- VF :Expl / Subject / Object
GENERATED OUTPUT
S NPSubj V det n sieht NPObj det n der Junge eine Ente
S Expl V NPObj NPSubj det n det n es sieht eine Ente der Junge
S NPObj V NPSubj det n det n eine Ente sieht der Junge
S Expl V NPSubj NPObj det n det n es sieht der Junge eine Ente

Figure 2: Generation of four word orders for *Der Junge sieht eine Ente*
Large plain arrows indicate inheritance links in the hierarchy. Smaller arrows indicate a crossing of fi nal classes which produces an output

tences with or without a temporal modifier, and for German sentences with or without an expletive *es*. This constitutes a sample of the annotated sentences generated by our hierarchy. Because of space constraints, we are not able to reproduce here the syntactic annotations, but the data is available upon request to the authors.

Figure 2 illustrates how we generate the German version of our sentence from Figure 1 (*the boy sees the duck*). We retain the English hierarchy, but add a special class for V2, called *Vorfeld* (at the left in Figure 2). Of course, the lexical items (at the right in both Figures 1 and 2) are different as well. This hierarchy correctly generates the four possible word-orders: *Der Junge sieht eine Ente*, *eine Ente sieht der Junge*, *es sieht der Junge eine Ente*, *es sieht eine Ente der Junge*. All the sentences have the same hypertag, except for the *Vorfeld* value, which is *Expletive*,*Subject*, or *Object*. Except for the classes in the lexicalization dimension, which are language dependent, classes are the same as for English, with the exception of the Vorfeld realization class and its descendants. (The figures omit some details for reasons of space.)

In using a single hierarchy for multiple languages, one of course is immediately faced with the question of what level of granularity one should use in expressing the categories of the metagrammar. For example, in describing a verb-second language such as German, one could opt to "bury" the syntax of verb-second in the language-specific partial description associated with the classes of the hierarchy. Alternatively, one could introduce (as we have done in Figure 2) a *Vorfeld* class which the German main-clause sentence (but not the English verb or the German embedded verb) obligatorily inherits. It is clear that this approach is interesting when several languages are to be modelled, some of which are verb-second, and others of which are not. Furthermore, one can might model the occupation of the Vorfeld either as subclasses of *Vorfeld* (as we have done), or using argument realization classes (in fact, the same as for English topicalization). Finally, if one is modelling a large number of V2 languages, one needs to consider the variety of V2 behavior found cross-linguistically with respect to the location of the non-finite verb (VO or OV), the occurrence of V2 in embedded clauses (see the complex behavior in Mainland Scandinavian), or the occasional V3 effect (Kashmiri). Thus, we see that our approach does not impose a linguistic choice for the cross-linguistic modelling of syntax, but it allows for a wide variety of approaches which can be chosen as a function of research and application needs.

4 Generating Cross-Framework Annotated Data

So far, we have explained how we produce sentences annotated for phrase-structure. We now explain how we produce other types of syntactic annotation: LFG F-structures, dependency annotations, and constraint grammar annotation.[13]

As we've seen in the previous section, the MG generator outputs pairs of <FeatureStructure-Tree>. When the MG is used to generate a grammar, then the tree is interpreted as a grammar rule (notion of "elementary tree" in the TAG framework) and Feature structure is meant to represent a description of this tree. However, the $< FeatureStructure - Tree >$ pair can be interpreted differently. In the previous section, we have seen that the trees generated by the MG may represent a whole Phrase-structure annotated sentence, and the feature structure represents a description of that sentence. But the generated tree does not have to represent a Phrase structure tree: it can as well represent a dependency tree, an LFG F-structure (with co-indices to account for reentrant features), or a flat tree consisting of only one root node and daughter leaves (with one leaf for each word in the sentence), where each leaf is annotated with constraint grammar annotation.

So, by using the the exact same MG hierarchy used for generating Phrase-structure annotation, we have generated the very same sentences for English and German, but this time annotated with dependency structures.

Figure 3 shows how we generate annotations for *The boy sees the duck*, for dependency. Note that the hierarchy is identical to the one on figure 1, except for the constraints for building the Quasi-tree. We use the same technique (changing the constraints on quasi-trees) to generate LFG F-structures, and constraint grammar annotations.

Of course, all the sentences in the appendix are generated in a similar fashion using 26 classes. The cross-language and -framework parallelism of the sentences is ensured by the hypertag. For instance, all the examples presented on figures 1, 2 and 3 have a common hypertag kernel:

$$\begin{bmatrix} \text{construction: Declarative} \\ \text{Sucat: Transitive} \\ \text{tense: simple english:+/-} \\ \text{german:+/-} \\ \text{Subject: NP} \\ \text{Object: NP} \end{bmatrix}$$

5 Conclusion and Future Work

We have shown a new way to produce cross-language and cross-framework test-suites, using a common MetaGrammar hierarchy, which reduces the need for human intervention in the development process. The same technique may be applied for generating cross-languages and cross-framework grammars, which is a problem we are currently investigating. We plan to expand the hierarchy to generate larger test-suites [14]

Moreover, in future work, we plan to apply the same technique to naturally occurring text, which would prove promising for producing parallel treebanks for various frameworks. If we assume that a treebank already exists (such as the Penn Treebank), our task would be to extract hypertags from the existing annotation (which, in the case of the Penn Treebank, can be done, using some heuristics), and then use the hypertags to generate annotations for the same corpus in different frameworks. If no treebank exists, we have to do the hypertagging by hand, and assign a dependency-style structure to the sentence. We can then proceed in the same manner as before, using a single hand-annotation to derive the treebanks for the different frameworks. Thus, we believe that the use of hypertags as a framework-neutral representation of relevant syntactic features can greatly reduce the dependence of treebanks on particular formalisms

[13]For the constraint grammar annotation, we have used the output for English of the ENCCG demo available online at www.connexor.com/demos.html. For LFG, and dependency annotations, we produce "standard" analysis, but as we have seen in the previous section, different linguistic choices could be made.

[14]Scaling up has not been a problem to generate grammars of more than 5000 rules (see (Kinyon and Prolo, 2002) for a detailed discussion), and presents no intrinsic diffi culties for test-suite generation.

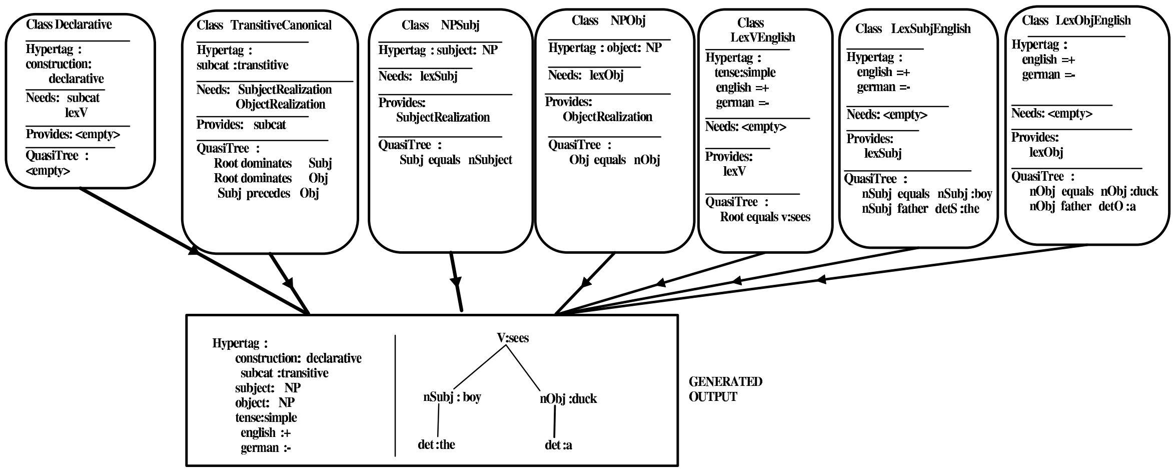

Figure 3: Generation of dependencies for *The boy sees a duck*

and frameworks.

References

A. Abeille, M. Candito, and A. Kinyon. 1999. FTAG: current status and parsing scheme. In *Proc. Vextal-99*, Venice.

M. Butt, H. Dyvik, T.H. King, H. Masuichi, and C. Rohrer. 2002. The parallel grammar project. In *proc. GEE-COLING*, Taipei.

M.H. Candito. 1996. A principle-based hierarchical representation of LTAGs. In *COLING-96*, Copenhagen.

M.H. Candito. 1999. *Representation modulaire et parametrable de grammaires electroniques lexicalisees*. Ph.D. thesis, Univ. Paris 7.

J. Carroll, E. Briscoe, and A. Sanfi lippo. 1998. Parser evaluation: a survey and a new proposal. In *LREC-98*, Grenada.

B. Gaiffe, B. Crabbe, and A. Roussanaly. 2002. A new metagrammar compiler. In *Proc. TAG+6*, Venice.

A. Kinyon and C. Prolo. 2002. A classifi cation of grammar development strategies. In *Proc. GEE-COLING*, Taipei.

A. Kinyon. 2000. Hypertags. In *COLING-00*, Sarrebrucken.

S. Lehman and al. 1996. Tsnlp — test suites for natural language processing. In *Proc. COLING-96*, Copenhagen.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English : the penn treeban. In *Computational Linguistics*, Vol 19.

J. Rogers and K. Vijay-Shanker. 1994. Obtaining trees from their description: an application to TAGS. In *Computational Intelligence 10:4*.

B. Srinivas. 1997. *Complexity of lexical descriptions and its relevance for partial parsing*. Ph.D. thesis, Univ. of Pennsylvania.

R. Sutcliffe, H. Koch, and A. McElligot, editors. 1996. *Industrial Parsing of Software Manuals*. Rodopi, Amsterdam.

F. Xia. 2001. *Automatic grammar generation from two perspectives*. Ph.D. thesis, Univ. of Pennsylvania.

Y.Mitsuishi, K. Torisawa, and T. Tsujii. 1998. HPSG-Style underspecifi ed japanese grammar with wide coverage. In *COLING-ACL-98*, Montreal.

Appendix 1: Sample of Parallel English-German sentences we generate.

Syntactic Phenomena	# Engl-Germ.	English	German
Declarative-NoModif-SimpleTense	1-4	The boy sees a duck	Der Junge sieht eine Ente
			Eine Ente sieht der Junge
			Es sieht der Junge eine Ente
			Es sieht eine Ente der Junge
Declarative-NoModif-CoumpoundTense	1-4	The boy has seen a duck	Der Junge hat eine Ente gesehen
			Eine Ente hat der Junge gesehen
			Es hat der Junge eine Ente gesehen
			Es hat eine Ente der Junge gesehen
Declarative-Modif-SimpleTense	2-12	Today the boy sees a duck	Heute sieht der Junge eine Ente
			Heute sieht eine Ente der Junge
		The boy sees a duck today	Der Junge sieht eine Ente heute
			Eine Ente sieht der Junge heute
			Es sieht der Junge eine Ente heute
			Es sieht eine Ente der Junge heute
			Der Junge sieht heute eine Ente
			Eine Ente sieht heute der Junge
			Es sieht heute der Junge eine Ente
			Es sieht heute eine Ente der Junge
			Es sieht der Junge heute eine Ente
			Es sieht eine Ente heute der Junge
Declarative-Modif-CoumpoundTense	2-12	Today the boy has seen a duck	Heute hat der Junge eine Ente gesehen
			Heute hat eine Ente der Junge gesehen
		The boy has seen a duck today	Der Junge hat eine Ente heute gesehen
			Eine Ente hat der Junge heute gesehen
			Es hat der Junge eine Ente heute gesehen
			Es hat eine Ente der Junge heute gesehen
			Der Junge hat heute eine Ente gesehen
			Eine Ente hat heute der Junge gesehen
			Es hat heute der Junge eine Ente gesehen
			Es hat heute eine Ente der Junge gesehen
			Es hat der Junge heute eine Ente gesehen
			Es hat eine Ente heute der Junge gesehen
SubordinateWithCompl-NoModif-SimpleTense	1-2	He says that the boy sees a duck	Er sagt, dass der Junge eine Ente sieht
			Er sagt, dass eine Ente der Junge sieht
SubordinateWithCompl-NoModif-CoumpoundTense	1-2	He says that the boy has seen a duck	Er sagt, dass der Junge eine Ente gesehen hat
			Er sagt, dass eine Ente der Junge gesehen hat
SubordinateWithCompl-Modif-SimpleTense	2-6	He says that today the boy sees a duck	Er sagt, dass heute der Junge eine Ente sieht
			Er sagt, dass heute eine Ente der Junge sieht
		He says that the boy sees a duck today	Er sagt, dass der Junge eine Ente heute sieht
			Er sagt, dass eine Ente der Junge heute sieht
			Er sagt, dass eine Ente heute der Junge sieht
			Er sagt, dass der Junge heute eine Ente sieht
SubordinateWithCompl-Modif-CoumpoundTense	2-6	He says that today the boy has seen a duck	Er sagt, dass heute der Junge eine Ente gesehen hat
			Er sagt, dass heute eine Ente der Junge gesehen hat
		He says that the boy has seen a duck today	Er sagt, dass der Junge eine Ente heute gesehen hat
			Er sagt, dass eine Ente der Junge heute gesehen hat
			Er sagt, dass eine Ente heute der Junge gesehen hat
			Er sagt, dass der Junge heute eine Ente gesehen hat
SubordinateNoCompl-NoModif-SimpleTense	2-6	He says the boy sees a duck	Er sagt der Junge sieht eine Ente
			Er sagt eine Ente sieht der Junge
			Er sagt es sieht der Junge eine Ente
			Er sagt es sieht eine Ente der Junge
SubordinateNoCompl-NoModif-CoumpoundTense	1-4	He says the boy has seen a duck	Er sagt der Junge hat eine Ente gesehen
			Er sagt eine Ente hat der Junge gesehen
			Er sagt es hat der Junge eine Ente gesehen
			Er sagt es hat eine Ente der Junge gesehen
SubordinateNoCompl-Modif-SimpleTense	2-12	He says today the boy sees a duck	Er sagt heute sieht der Junge eine Ente
			Er sagt heute sieht eine Ente der Junge
		He says the boy sees a duck today	Er sagt der Junge sieht eine Ente heute
			Er sagt eine Ente sieht der Junge heute
			Er sagt der Junge sieht heute eine Ente
			Er sagt eine Ente sieht heute der Junge
			Er sagt es sieht der Junge eine Ente heute
			Er sagt es sieht eine Ente der Junge heute
			Er sagt es sieht der Junge heute eine Ente
			Er sagt es sieht eine Ente heute der Junge
			Er sagt es sieht heute der Junge eine Ente
			Er sagt es sieht heute eine Ente der Junge
SubordinateNoCompl-Modif-CoumpoundTense	2-12	He says today the boy has seen a duck	Er sagt heute hat der Junge eine Ente gesehen
			Er sagt heute hat eine Ente der Junge gesehen
		He says the boy sees a duck today	Er sagt der Junge hat eine Ente heute gesehen
			Er sagt eine Ente hat der Junge heute gesehen
			Er sagt der Junge hat heute eine Ente gesehen
			Er sagt eine Ente hat heute der Junge gesehen
			Er sagt es hat der Junge eine Ente heute gesehen
			Er sagt es hat eine Ente der Junge heute gesehen
			Er sagt es hat der Junge heute eine Ente gesehen
			Er sagt es hat eine Ente heute der Junge gesehen
			Er sagt es hat heute der Junge eine Ente gesehen
			Er sagt es hat heute eine Ente der Junge gesehen