Toward Synchronous Extensible Dependency Grammar

Michael Gasser
School of Informatics and Computing
Indiana University
Bloomington, Indiana USA 47405
gasser@cs.indiana.edu

Abstract

Extensible Dependency Grammar (XDG; Debusmann, 2007) is a flexible, modular dependency grammar framework in which sentence analyses consist of multigraphs and processing takes the form of constraint satisfaction. This paper shows how XDG lends itself to grammar-driven machine translation and introduces the machinery necessary for synchronous XDG. Since the approach relies on a shared semantics, it resembles interlingua MT. It differs in that there are no separate analysis and generation phases. Rather, translation consists of the simultaneous analysis and generation of a single source-target "sentence".

1 Introduction

Despite the impressive advances in statistical machine translation (SMT) in the last 20 years, rule-based machine translation remains appropriate when the goal is publication quality translation of documents, especially within a restricted domain (Ranta et al., 2010). Furthermore, SMT is ruled out when bilingual corpora are unavailable for the language pairs of interest.

Our long-term goals are translation between English and the Ethiopian languages Amharic and Oromo and between Spanish and the Andean language Quechua within several restricted domains related to science and health. In addition to the language-specific software that we are developing, all of it freely available, we plan to develop a range of open-source tools that will enable other developers to take on translation projects of this sort involving under-resourced languages.

In search of a general purpose grammatical framework, we settled on Extensible Dependency Grammar (XDG), developed by Ralph Debusmann and colleagues (Debusmann et al., 2004; Debusmann, 2007), because of its modular structure; its extensibility; and its simple, declarative format. Dependency grammars have attracted considerable attention within computational linguistics in recent years due to their simplicity, the ease of the integration of syntax and semantics, and their handling of word-order variation and long-distance dependencies. These advantages apply to rule-based machine translation as well, and dependency grammar is increasingly seen as a viable and productive framework for RBMT (see, for example, Bick, 2007; Čmejrek et al., 2003; Diaconescu, 2004; Mel'čuk and Wanner, 2006). Among the various dependency grammar frameworks, XDG has the disadvantage that it has not been tested with wide coverage grammars and unconstrained input (see Bojar (2005) for an initial attempt at this task for Czech). However, we feel that the XDG's flexibility and its proven capacity to handle complex syntactic constraints outweigh this drawback, especially since our goal is relatively small grammars for restricted input.

This paper proposes a way to integrate translation into XDG. After a brief overview of XDG, Section 3 shows how it is possible to accommodate translation within the framework relatively

F. Sánchez-Martínez, J.A. Pérez-Ortiz (eds.)
Proceedings of the Second International Workshop on Free/Open-Source Rule-Based Machine Translation, p. 3–10
Barcelona, Spain, January 2011. http://hdl.handle.net/10609/5643

simply with the addition of cross-lingual links between elements in the lexicons of the two languages. Section 4 illustrates the proposal with a simple example based on small grammar fragments for English and Amharic. Section 5 discusses the current status of our project. Section 6 concludes with a consideration of ongoing work.

2 Extensible Dependency Grammar

2.1 Dimensions and principles

Like other dependency grammar frameworks, XDG is lexical; the basic units are words and the directed, labeled arcs connecting them. In the simplest case, an analysis ("model" in XDG terms) of a sentence is a weakly connected, directed graph over a set of **nodes**, one for each word in the analyzed sentence and a distinguished **root node** representing the end-of-sentence punctuation. As in some other dependency frameworks, XDG permits analyses at multiple levels, known as **dimensions**, each corresponding to some level of grammatical abstraction. For example, one dimension could represent syntax, another semantics. Two dimensions may also be related by an explicit interface dimension which has no arcs itself but constrains how arcs in the related dimensions associate with one another.

In the general case, then, an analysis of a sentence is a **multigraph**, consisting of a separate dependency graph for each dimension over a single sequence of word nodes. For most languages, syntax requires at least two dimensions, an immediate dominance dimension responsible for syntactic relations, including agreement, and a linear precedence dimension responsible for word order. For the sake of simplicity, the syntactic dimensions are collapsed into a single syntax dimension (SYN) in this paper. Likewise for semantics, multiple dimensions may be necessary, but in this paper we confine ourselves to a single one (SEM), corresponding closely to Debusmann's predicate argument dimension.

Figure 1 shows a possible analysis for the English sentence *the water is contaminated* on these two dimensions. Arrows go from heads to dependents (daughters) in the figure. On the SEM dimension, we maintain the convention that only content words participate in the representation. That is, any grammatical words appearing in the SYN dimension are effectively "deleted" in the SEM dimension. Actual deletion is not possible because of the constraint that the graph on each dimension be connected. As shown in the figure, "virtual deletion" is handled in XDG through the use of special del arcs (Debusmann, 2007).

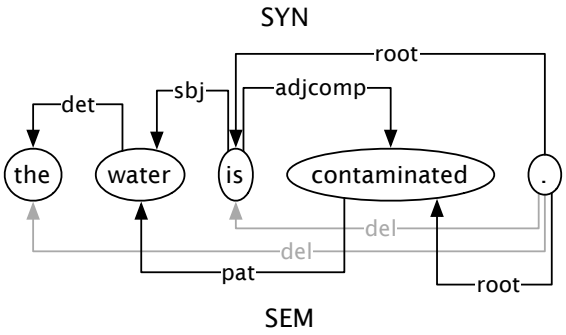

Figure 1: XDG analysis of an English sentence.

A grammatical analysis is one that conforms to a set of constraints, each generated by one or another **principle**. Each dimension has its own set of principles, and all principles are defined in a language-independent fashion. Some examples:

- Principles concerned with the structure of the graph; for example, the Tree Principle, which constrains the graph to be a tree
- The Valency Principle, governing the labels on the arcs into and out of nodes
- The Agreement Principle, constraining how features of certain words must match features of other words
- The Agr Principle, constraining what values the agreement features of a word can have
- The Order Principle, concerned with the order of the words in the sentence
- The Linking End Principle, an interface dimension principle that associates arc labels on one of the dimensions with arc labels on the other.

2.2 The lexicon

An XDG grammar of a language consists of a set of dimensions, each with its own set of principles and arc labels, and a lexicon. As XDG is com-

pletely lexical, it is at the level of words that the principles apply. That is, all specific grammatical constraints are stored in word-level units.

The **lexicon** consists of a set of **entries** arranged in an inheritance hierarchy. At their most specific, entries are associated with particular wordforms such as *contaminated*. Higher up in the hierarchy are entries associated with lexemes such as CONTAMINATE. At the top of the hierarchy are entries associated with lexical classes such as verb (V).

Each entry specifies the more abstract entries that it inherits from, if any, and one or grammatical constraint specifications. A grammatical constraint specification has the following form.

dimension:

principle: *constraint attributes*

For example, consider the constraints that participate in the Valency Principle. Entry 1 shows a portion of the English transitive verb entry. The entry includes a pointer to one parent entry (V) and three valency constraints on the SYN dimension. The word requires outgoing subject (sbj) and object (obj) arcs and an incoming root arc.[1] (The "!" represents the requirement of exactly one arc with the given label.)

Entry 1 English transitive verb

```
- name: V_T
  classes: [V]
  syn:
    val: {out: {sbj: !, obj: !},
          in: {root: !}}
```

The overall structure of a portion of the lexicon is shown in Figure 2. Each rectangle with a double border represents a lexical entry; the arrows represent inheritance relations.

2.3 Processing

Parsing within XDG takes the form of constraint satisfaction. Given an input sentence to be analyzed, **lexicalization** creates a node for each word in the sentence and searches the lexicon for entries that match the words. A copy of each matching entry (a **node entry**) is added to the nodes; all of the information in ancestors of the matching nodes in the lexical inheritance hierarchy is also

[1] In our simple grammar, there are no dependent clauses so all finite verbs are the heads of sentences.

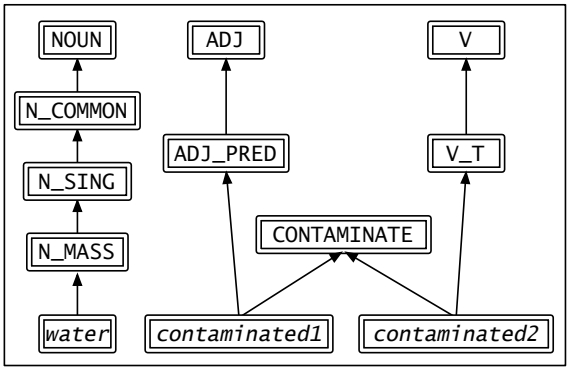

Figure 2: Portion of the English lexicon.

copied to the node entries. Each node is identified by an index representing its position in the input sentence.

For morphologically complex languages, such as most of those we are concerned with, it is impractical to store all wordforms in the lexicon. In Gasser (2010), we showed how a morphological analyzer can be incorporated in sentence analysis in XDG. Morphological analysis of the input words results in a lexeme and a set of grammatical features for each analyzed word. We have developed finite state morphological analyzers for Amharic, Oromo, and Quechua for this purpose; the theoretical approach behind the analyzers is described in Gasser (2009).

Next, lexicalization invokes the principles that are referenced in the matching lexical entries and their ancestors in the lexical hierarchy. Each of these principle invocations results in the instantiation of one or more constraints, each applying to a set of variables. For example, each node n has a daughters variable whose value is the set of indices of the daughter nodes of n. Among the constraints that apply to such a variable are those associated with the Tree Principle.

For ambiguous words, lexicalization finds multiple entries in the lexicon. The result is that some nodes may end up with more than one node entry, each with its own set of constraints. Each node's entries take the form of a list. A **disambiguation variable** is created for each node; the value of this variable is the index of an entry in the node entry list that satisfies all of the grammatical constraints for that node.

Finally, constraint satisfaction is applied to the variables and constraints that have been instantiated. If this succeeds, it returns all possible complete variable assignments, each corresponding to a single analysis of the input sentence, that is, a multigraph across the sentence nodes. Lexical disambiguation occurs during constraint satisfaction when the values of disambiguation variables for nodes are restricted by the different constraints.

Because an XDG grammar is declarative, it can be used for generation as well as for analysis. The main difference for generation is that the semantic input does not specify the positions for words in the output. This problem can be handled in a straightforward fashion through the creation of a position variable for each node; these variables are constrained by the Order Principle.

The XDG framework has been applied to parsing for a number of languages but, to our knowledge, never to translation. In what follows we show how the modularity and application of constraint satisfaction that characterize XDG lend themselves to translation.

3 Multiple languages in XDG

3.1 Semantics

Assume we have XDG grammars in the form of hierarchical lexicons for two or more languages. Each grammar specifies constraints on one or more syntactic dimensions and one or more semantic dimensions. A single interface dimension relates syntax to semantics, constraining the way in which arc labels on one syntactic dimension are associated with arc labels on one semantic dimension. The grammars for the different languages share their semantics, in the sense that a particular class of predicate types has a fixed representational format on the semantic dimension in each grammar. In the semantic representation for the sentence shown in Figure 1, the semantic head of the sentence is the word *contaminated*, which has the single pat (patient) argument *water*.

3.2 Syntax

Now consider the analysis of one possible Amharic translation of the same sentence, shown in Figure 3. The Amharic sentence consists of only two words, the noun ውሃው *wihaw* 'the water'[2] and the verb ተበክሏል *tɛbɛkkɨlwal* '(it) has been contaminated'. The graph on the semantic dimension is identical to that for the English sentence, except that the nodes are associated with completely different words and the nodes corresponding to the grammatical words in the English sentence, *the* and *is*, are missing.

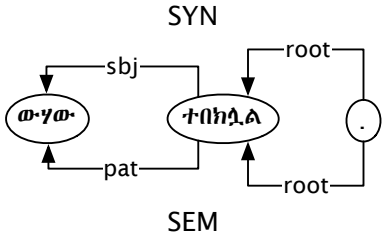

Figure 3: XDG analysis of an Amharic sentence.

It is in the syntax-semantics interface dimension (SYNSEM) that the differences between the two languages are spelled out. For English the word *contaminated* is ambiguous: here we consider only its interpretation as a predicate adjective, which happens to be the right choice for this sentence. For this case, the word inherits from Entry 2.

Entry 2 English predicate adjective

```
- name: ADJ_PRED
  syn:
    val: {in: {adjcomp: !}}
  sem:
    val: {in: {root: !}, out: {pat: !}}
  synsem:
    linkend: {pat: [adjcomp]}
```

According to this entry, a word belonging to this lexical class must have an incoming adjective complement adjcomp arc on the SYN dimension. On the SYNSEM dimension, there is constraint for the Linking End Principle, specifying that a pat daughter of this word on the SEM dimension must be an adjcomp daughter of some node on the SYN dimension.

For Amharic the corresponding entry is quite different. The word ተበክሏል is not an adjective but a passive verb in the present perfect tense. For

[2]The definite article in Amharic takes the form of a noun suffix.

Amharic intransitive change-of-state verbs such as ተበከለ 'be contaminated', the present perfect form can refer to the state resulting from the change of state, as it does in this case. For this sentence the verb inherits from Entry 3. According to this entry, a word belonging to this lexical class must have an incoming root arc and an outgoing sbj arc on the SYN dimension. On the SYNSEM dimension, the linking end constraint associates the semantic pat with the syntactic sbj.

Entry 3 Amharic intransitive present perfect verb

```
- name: V_I_PRESPERF
  syn:
    val: {in: {root: !}, out: {sbj: !}}
  sem:
    val: {in: {root: !}, out: {pat: !}}
  synsem:
    linkend: {pat: [sbj]}
```

3.3 Cross-lingual links

Given lexicons for two languages with common semantics, we associate entries in one language with corresponding entries in the other using links that behave like the inheritance links within each lexicon. Three of these links are shown in Figure 4.

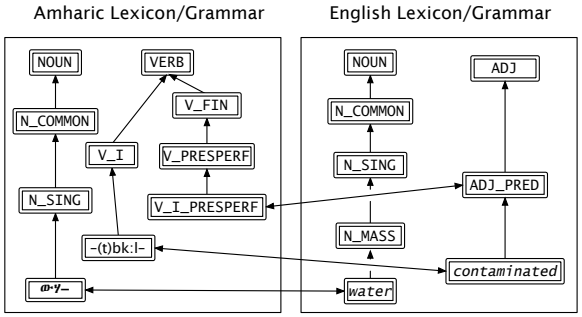

Figure 4: Partial lexicons of two languages with three cross-lingual links.

One link joins the entry for the English word *water* with that for the Amharic lexeme ው·ሃ. Another joins the entry for the English adjective *contaminated* with that for the Amharic verb lexeme *-(t)bk:l-* 'be contaminated'. A third joins the entry for English adjective class ADJ_PRED with that for the Amharic verb class V_I_PRESPERF. Note that these are not the only possible links out of or into these entries. For example, like English, Amharic has predicate adjectives, so the English ADJ_PRED entry would also be associated with the Amharic ADJ_PRED entry.

With these simple inheritance links in place, most translation is little more than parsing. We describe this process in the next section.

4 Translation in XDG

The key idea behind synchronous XDG is to view the dimensions associated with different languages as part of one big multilingual grammar, which includes the SEM dimension that is shared by all of the languages. On this view a sentence and its translation into one or more other languages are just a multigraph connecting the nodes of the sentence. This is illustrated for our example sentence in Figure 5; for simplicity we have omitted the semantic dimension. Note that the positions of the nodes in the target language (numbers in the small rectangles) are a part of the representation.

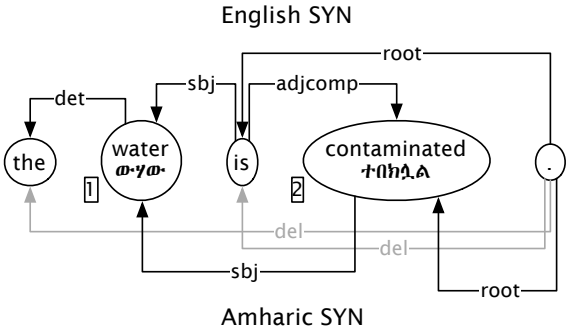

Figure 5: Multigraph representation of a bilingual "sentence".

Beginning with the English sentence as input, translation into the Amharic sentence is just ordinary XDG constraint satisfaction with the additional step of morphological generation of the Amharic words at the end of the process. In what follows, we examine each step in the translation of the English sentence *the water is contaminated*.

4.1 Lexicalization

As with ordinary sentence analysis, each word is assigned a node. Since the source language in this case is English, the morphological analysis step is

skipped. For each node all matching entries in the lexicon are copied, resulting in node entries, and the principles referenced in these entries are invoked, resulting in constraints. The word *contaminated* is ambiguous so multiple node entries are created for node 4. A disambiguation variable is also created for each node. At this point a number of syntactic and semantic constraints have been instantiated, including those associated with various principles: Valency, Agreement, Agr, Order, Tree, and LinkingEnd.

Related entries in the Amharic lexicon are accessed via the cross-lingual links, with lexicalization operating as before. That is, the principles in the entries found in the Amharic lexicon are also invoked, resulting in constraints relevant to the Amharic SYN and SYNSEM dimensions. In addition, the lexeme forms associated with the Amharic entries are copied to the node entries. Figure 6 depicts node 4 at this point. The small ovals represent node entries for two different senses of *contaminated*. Both of these entries are associated with the Amharic lexeme *-(t)bk:l-* 'be contaminated'. The instantiated constraints are indicated by arrows. These include English syntactic, Amharic syntactic, and semantic constraints related to the different senses of *contaminated*. The disambiguation variable for the node, 4disambig, has also been created.

The Amharic verb root *-(t)bk:l-* is not a pronounceable surface form; the surface form will have to be generated at the end of processing, once the grammatical features of the word are known. Both node entries inherit a grammatical constraint specification from the Amharic verb entry that is associated with the Agr Principle and will provide the required grammatical features once constraint satisfaction has taken place. This specifies that an Amharic verb must take one of eight possible person-number-gender combinations as the value of its subject agreement feature. A variable has been created for this agreement feature (`?4amsyn_agr_sbj` in the figure). Similarly, node 2 (*water*) has an agreement variable for the Amharic noun, which must get a value for its definiteness feature.

Two words in the input sentence correspond to nothing in the Amharic translation, *the* and *is*. Recall, however, that the same set of nodes must

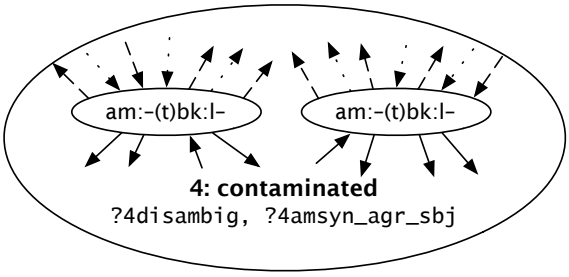

Figure 6: A node after lexicalization.

be present on every dimension in an XDG analysis. For these words, there are cross-lingual links to Amharic entries representing "empty nodes". On the Amharic SYN dimension, these nodes are constrained to have incoming del arcs. For more on empty nodes in XDG, see Gasser (2010) and Pelizzoni and Nunes (2005).

4.2 Constraint satisfaction and morphological generation

Constraint satisfaction searches for variable assignments that satisfy all of the constraints that were instantiated during lexicalization. Among the variables are ones for the daughters and mothers of all nodes on each dimension, ones for the daughters and mothers of all nodes specific to particular arc labels on each dimension, ones for the positions of the words in the target language sentence, ones for the agreement features of nouns and verbs, and ones for lexical disambiguation for each node.

Consider how node 4 ends up with a sbj arc to node 2 on the Amharic SYN dimension. Based on the initial constraints, node 4 expects an outgoing sbj arc, and node 2 expects an incoming sbj or obj arc on the Amharic SYN dimension. Since node 2 is the only node that can satisfy node 4's constraint, both constraints are satisfied by binding the variable `?4amsyn_sbj_out` to $\{2\}$ and the variable `?2amsyn_sbj_in` to $\{4\}$.

A multigraph is created for each variable assignment. The positions of words in the output sentence are read off of the values of the target-language position variables for each node.

When the target language is morphologically complex, as in this case, each of the output word forms for lexical nodes must be generated from

the root and the values of the agreement variables for the node and language. As well as analyzers, we have implemented finite state morphological generators for the morphologically complex languages in our project, Amharic, Oromo, and Quechua; these are invoked when one of these languages is the target language of translation. For node 4 in the example sentence, the Amharic root is *-(t)bk:l-*, the value of the subject feature variable is third person singular masculine, and the value of the tense-aspect-mood feature variable is present perfect. Given these inputs to the Amharic morphological generator, the wordform ተበክሏል *tɛbɛkkɨlwal* is output for node 4. For node 2, the Amharic root is ውሃ *wɨha*, and the value of the definiteness feature variable is True.[3] The generator outputs the form ውሃው *wɨhaw*.

4.3 Ambiguity and disambiguation

As noted in Section 2.3, lexical disambiguation in XDG takes the form of the constraining of a node's disambiguation variable during constraint satisfaction. Ambiguity that cannot be resolved within a sentence results in multiple analyses.

In a translation context, ambiguous source language words are often associated with completely different target language words. For example, the English noun *water* corresponds to the Amharic noun ውሀ while the English verb *water* corresponds to the Amharic object+verb phrase ውሀ አጠጣ, literally 'cause to drink water'. Within the translation framework described above, source language analysis and target language generation take place simultaneously, so a source language word such as *water* will be disambiguated while the target language constraints are being sorted out. It is possible that it would be more efficient to deal with the source language ambiguity first, as far as this is possible. Within the XDG framework, one alternative would be to run constraint satisfaction first on the constraints applying to source language dimensions and only later to run it on the target language dimensions. We have not yet experimented with this possibility.

More challenging is ambiguity that is only apparent in the context of translation, that is, one-to-many source-to-target ambiguity. Sometimes disambiguation is possible in such cases on the basis of other words in the source language sentence. For example, the English verb *break* corresponds to several Amharic verbs, depending on the type of breaking action, and the action may be inferred from the patient of the breaking. If the patient is *string*, the appropriate Amharic verb is በጠሰ, whereas if the patient is *glass*, the Amharic translation is ሰበረ. For now we treat this sort of cross-lingual ambiguity as we treat intra-language ambiguity; we simply create a new candidate node entry for each possibility. This leaves the ambiguity unresolved in all cases. One way to handle this sort of ambiguity is to create a database of associations between source-language contextual elements and target-language alternatives that is independent of the XDG lexicons and the usual constraint satisfaction mechanism. The associations would be invoked before or during constraint satisfaction to facilitate selection of one of the target-language alternatives. We are currently exploring this possibility.

5 Implementation

Because we intend in the long run to make some basic modifications to XDG, in particular to add weights to the constraints, we have re-implemented XDG from the bottom up, remaining mostly faithful to constraint satisfaction in the Mozart/Oz constraint programming framework that XDG relies on. Our implementation is in Python; lexicon/grammars are encoded in YAML format.

All of our software, including the XDG implementation, lexicons, and morphological analyzers and generators for Amharic, Spanish, Oromo, and Quechua, is available under a GNU GPL3 license, at `http://www.cs.indiana.edu/~gasser/Research/software.html`.

6 Conclusions

This paper has attempted to show how the properties of XDG, especially the multi-dimensionality and modularity, lend themselves to grammar-driven machine translation. A synchronous XDG grammar consists of modular lexicons of two or more languages and a set of simple cross-lingual

[3] For simplicity we omit a number of other features required for the morphological generation of Amharic verbs and nouns.

links joining lexical entries. Source and target language sentences are viewed as a single abstract "sentence" consisting of bilingual nodes joined by arcs on dimensions at various levels of abstraction. This representation allows us to capitalize on what is shared between the languages as well as to keep separate what is different. Furthermore, because lexicons are represented declaratively, the knowledge required for translation in one direction suffices for translation in the opposite direction.

This approach has more in common with traditional interlingua approaches than with transfer approaches, the SEM dimension playing the role of the interlingua. However, unlike in usual interlingua translation, there are no separate analysis and generation phases (except for morphological analysis or generation, when these are necessary). Constraint satisfaction seeks a representation of the sentence on all levels simultaneously: semantics constrains syntax, syntax constrains semantics, and, through semantics, the evolving syntactic representations of source and target languages constrain one another. It is quite possible, for example, for some of the target-language syntax to take shape before the source-language syntax is sorted out.

The contribution of this paper is a theoretical proposal. It offers no evaluation data at all. Thus the framework is really only the first step towards the development of the machine translation system we envision. Our next step is to test it on a much wider variety of sentence types, including the full range of possible syntactic differences between languages (Mel'čuk and Wanner, 2006).

References

Bick, E. (2007). Dan2eng: wide-coverage Danish-English machine translation. In *Proceedings of Machine Translation Summit Xi*, pages 37–43, Copenhagen.

Bojar, O. (2005). Problems of inducing large coverage constraint based dependency grammar. In et al., H. C., editor, *Constraint Solving and Language Processing, First International Workshop, CSLP 2004*, pages 90–103, Berlin. Springer Verlag.

Debusmann, R. (2007). *Extensible Dependency Grammar: A Modular Grammar Formalism Based On Multigraph Description*. PhD thesis, Universität des Saarlandes.

Debusmann, R., Duchier, D., and Kruijff, G.-J. M. (2004). Extensible dependency grammar: A new methodology. In *Proceedings of the COLING 2004 Workshop on Recent Advances in Dependency Grammar*, Geneva/SUI.

Diaconescu, S. (2004). Multiword expression translation using generative dependency grammar. In *Proceedings of EsTAL*, Alicante, Spain.

Gasser, M. (2009). Semitic morphological analysis and generation using finite state transducers with feature structures. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 309–317, Athens, Greece.

Gasser, M. (2010). A dependency grammar for Amharic. In *Proceedings of the Workshop on Language Resources and Human Language Technologies for Semitic Languages*, Valletta, Malta.

Mel'čuk, I. and Wanner, L. (2006). Syntactic mismatches in machine translation. *Machine Translation*, 20(2):81–138.

Pelizzoni, J. M. and Nunes, M. d. G. V. (2005). N:m mapping in XDG — the case for upgrading groups. In *Proceedings of the Workshop on Constraint Solving and Language Processing*, Roskilde, Denmark.

Ranta, A., Angelov, K., and Hallgren, T. (2010). Tools for multilingual grammar-based translation on the web. In *Proceedings of the Association for Computational Linguistics System Demonstrations*, Beijing.

Čmejrek, M., Cuřín, J., and Havelka, J. (2003). Czech-english dependency-based machine translation. In *Proceedings fo the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 83–90, Budapest.