Toward a Rule-Based System for English-Amharic Translation

Michael Gasser

Indiana University
Bloomington, Indiana, USA
gasser@indiana.edu

Abstract

We describe key aspects of an ongoing project to implement a rule-based English-to-Amharic and Amharic-to-English machine translation system within our L^3 framework. L^3 is based on Extensible Dependency Grammar (Debusmann, 2007), a multi-layered dependency grammar formalism that relies on constraint satisfaction for parsing and generation. In L^3, we extend XDG to multiple languages and translation. This requires a mechanism to handle cross-lingual relationships and mismatches in the number of words between source and target languages. In this paper, we focus on these features as well as the advantages that L^3 offers for handling structural divergences between English and Amharic and its capacity to accommodate shallow and deep translation within a single system.

1. Rule-Based Machine Translation

Among other disadvantages, African languages (and the communities of people who speak them) suffer from a lack of available documents in the languages, one aspect of the Linguistic Digital Divide (Paolillo, 2005). Machine translation (MT) and computer-assisted translation (CAT) could play a role in alleviating this problem. The ultimate goal of our project is the application of MT and CAT to the production of publication-quality documents in the major languages of the Horn of Africa.

Despite the impressive recent achievements of statistical machine translation (SMT), rule-based machine translation (RBMT) continues to offer advantages in certain contexts (Barreiro et al., 2011; Bond et al., 2011; Forcada et al., 2011; Mayor et al., 2011; Mel'čuk and Wanner, 2006; Ranta et al., 2010). SMT requires large parallel corpora, which are not available for under-resourced languages. The data sparsity problem is especially serious for morphologically complex languages such as Amharic because such languages have very large numbers of distinct word forms. Finally, for SMT systems, which normally rely on relatively primitive models of constituent order, significant structural differences between the languages can present problems.[1]

The relative advantages of SMT and RBMT also depend on the purpose of the MT system. For extracting the gist from a document, SMT systems already perform adequately for some language pairs and some domains. When the goal is publication-quality documents, however, an RBMT or RBCAT system, designed for a narrow content domain, would perform better (Ranta et al., 2010).

For our purposes, RBMT is clearly the way to start, and we are developing a framework for RBMT and RBCAT systems for under-resourced languages, L^3. L^3 relies on a powerful and flexible grammatical theory that we hope will be able to handle arbitrary syntactic divergences between languages. At the moment, we are far from our long-term goal of a set of tools that would allow developers to rapidly design MT systems for limited domains in a new language pair, something analogous to what Apertium (Forcada et al., 2011) offers. In this paper we discuss some features of L^3 that have emerged out of the development of a small prototype English-Amharic translation system.

In the next section, we introduce Extensible Dependency Grammar (XDG), the grammatical formalism behind L^3. Next we discuss the enhancements to XDG that are required for MT, including two aspects of the system not described in our previous work (Gasser, 2011b), the implementation of both shallow and deep MT and the handling of mismatches in the number of words. Next we show how L^3 deals with structural divergences between languages. We conclude with a discussion of ongoing work.

2. Extensible Dependency Grammar

Dependency grammars are a popular alternative to constituency grammars because of 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 RBMT as well (see, for example, Bick, 2007; Mel'čuk and Wanner, 2006). XDG (Debusmann et al., 2004; Debusmann, 2007) is a flexible, modular dependency grammar formalism that relies on constraint satisfaction for processing. Although XDG has not been tested with wide coverage grammars and unconstrained input, we believe that its flexibility and its proven capacity to handle complex syntactic constraints outweigh this drawback, especially since our goal is relatively small grammars for restricted input. Debusmann (2007) has developed a partial XDG grammar of English that handles many complex syntactic phenomena, and in earlier work (Gasser, 2010), we have shown how the unusual features of Amharic relative clause syntax can also be dealt with in this framework.

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 of a sentence is a directed graph over a set of **nodes**, one for each word in the analyzed sentence, including a distinguished **root node** representing the end-of-sentence punctuation. As in some other dependency frameworks, for example, Meaning-Text Theory (see Mel'čuk and Wanner, 2006 for the application of MTT to MT), XDG permits analyses on

[1] Hybrid RBMT-SMT systems are beginning to address some of these deficiencies.

multiple **dimensions**, each corresponding to some level of grammatical abstraction.
In L^3, each language is represented on two dimensions, Immediate Dominance (ID) and Linear Precedence (LP), and a further dimension, Semantics (SEM), is responsible for language-independent conceptual structure. A key feature of XDG is the **interface dimensions** that relate dimensions such ID and LP to one another. Interface dimensions have no arcs of their own but instead constrain how arcs in the related dimensions correspond to one another.

2.1. Analyses as multigraphs

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. Figure 1 shows a possible analysis for the English sentence *the doctor cured the patient* on the ID and SEM dimensions, each represented by a plane in the figure. (The LP dimension is omitted here and in subsequent figures for the sake of simplicity.) Each node is represented by a pair of circles or squares joined by dashed lines. The square node is the end-of-sentence root node.[2] Arrows go from heads to dependents (daughters). For simplicity, we refer to the core arguments of semantic predicates as arg1 and arg2, rather than agent, patient, etc. On the SEM dimension, we maintain the convention that only content words participate in the representation. That is, any strictly grammatical words appearing in the ID dimension are effectively "deleted" in the SEM dimension. As shown in the figure, "virtual deletion" is handled in XDG through the use of special del arcs (Debusmann, 2007). In subsequent figures, we omit the del arcs, indicating deleted nodes with unfilled circles.
A grammatical analysis is one that conforms to a set of **constraints**, each applying to a particular dimension. Constraints belong to several categories, the most important of which are **graph** constraints, restricting the structure of the dependency graph; **valency** constraints, governing the labels on the arcs into and out of nodes; **agreement** constraints; **order** constraints; and various **linking** constraints that apply to interface dimensions and govern the manner in which arcs on one dimension are associated with arcs on another dimension.

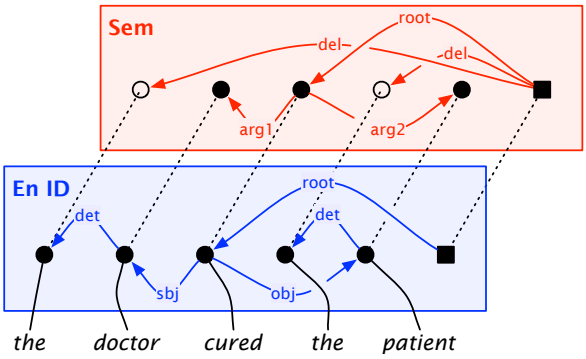

Figure 1: XDG analysis of an English sentence.

[2]XDG does not handle sentence fragments; we are currently extending the system to have this capacity.

2.2. The lexicon

An XDG grammar of a language consists of a set of dimensions, each with its own set of constraints and arc labels, and a lexicon. As XDG is completely lexical, all specific grammatical constraints are stored in word-level units.
The **lexicon** consists of a set of **entries** associated with word forms or lemmas. Each entry contains one or more grammatical constraint attributes. Entry 1 shows a portion of the entry for the English lemma *cure*. The entry includes three valency constraint attributes on the ID dimension. The word requires outgoing subject (sbj) and object (obj) arcs and an incoming root arc.[3] (The "!" represents the requirement of exactly one arc with the given label.)

Entry 1 Portion of English entry for the lemma *cure*

```
- lemma: cure
  ID:
    out: {sbj: !, obj: !}
    in: {root: !}
```

2.3. Parsing and generation

Parsing within XDG begins with a **lexicalization** phase which creates a node for each word in the sentence and searches the lexicon for matching entries. For morphologically complex languages, such as Amharic, it is impractical to store all word forms in the lexicon, and we employ in-house morphological analyzers to pre-process the input words for such languages. Morphological analysis of the input words results in one or more lemmas and sets of grammatical features for each analyzed word. The word forms or lemmas in the input are matched against lexical entries, and a copy of each matching entry (a **node entry**) is added to the nodes. Each node is identified by an index representing its position in the input sentence.
The next phase is **variable and constraint instantiation**. Each of the constraints referenced in the constraint attributes in the node entries is instantiated. The constraints apply to a set of variables, which are created during this phase. 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 graph constraints.
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.
Because an XDG grammar is declarative, it can be used for generation as well as for analysis. The main difference is that for generation the semantic input does not specify the positions for words in the output. This problem is handled in a straightforward fashion through the creation of a position variable for each node; these variables are constrained by explicit order constraints. Another difference relates to the frequent mismatch in the number of nodes between semantics and the ID and LP dimensions (Pelizzoni and Nunes, 2005), a problem we discuss below.

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

3. L^3

L^3 is an extension of XDG to translation. Enhancements to the basic framework include separate lexica for each language and **cross-lingual links** joining lexical entries in different lexica. Within L^3 we treat semantics, consisting of a single SEM dimension, as a language with its own lexicon.

3.1. Multilingual multigraphs

Adaptation of XDG to translation is straightforward once we make the leap to thinking of a sentence and its translation into another language as a single multilingual "sentence" with a single set of multilingual word nodes. This is illustrated for the sentence of Figure 1 in Figure 2, where we have shown the ID dimensions for English and Amharic and the SEM dimension.[4] Note that the order of the words in the Amharic output is subject-object-verb rather than subject-verb-object, as in the English input.

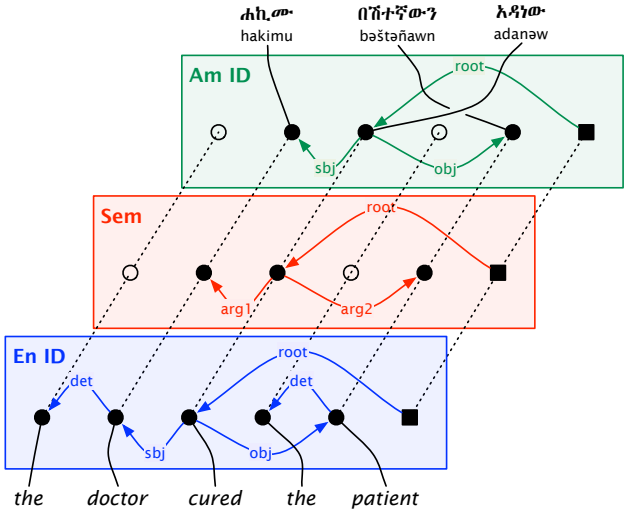

Figure 2: Multigraph for a bilingual "sentence".

3.2. Cross-lingual links

In keeping with the lexical nature of XDG, all cross-lingual knowledge in the system takes the form of links joining lexical entries in different languages. By convention in L^3, these always join the ID dimension of a language to another dimension, either the SEM dimension of semantics, or the ID dimension of another language. These links specify a target lexical entry and optionally a constraint attribute on the interface dimension joining the two arc dimensions. Entry 2 shows the portion of the English entry for the verb *cure* that links it to the entry in the semantics lexicon for the corresponding CURE event type and the corresponding entry in the Amharic lexicon, አዳነ *adanə*. There are attributes for the linking constraint, linking end, on both interface dimensions, IDSEM and IDID.

The linking end constraint is illustrated in general and for the special case of IDSEM for the subject of *cure* in Figure 3. For a node n, linking end specifies a relationship between arc labels $l1$ and $l2$ on dimensions $d1$ and $d2$ respectively. It constrains n to have as the daughter on its $l1$ arc in $D1$ a node which is the daughter of some node on $D2$, not necessarily n, on an arc with label $l2$. As we will see below, linking end, and other similar linking constraints, are the key to representing structural differences on different dimensions (ID and Sem or ID in one language and ID in another) across the same set of word nodes.

Entry 2 Cross-lingual links in the entry for *cure*

```
- lemma: cure
  cross:
    sem:
      lex: CURE
      IDSem:
        linkend: {arg1: [sbj]}
    am:
      lex: adane
      IDID:
        linkend: {sbj: [sbj]}
```

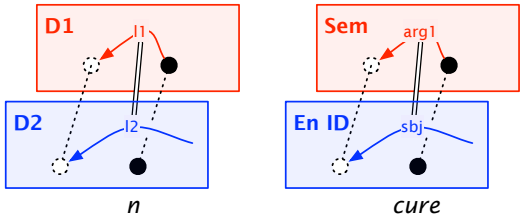

Figure 3: Linking end, in general and for subject of *cure*.

3.3. Shallow and deep translation

The fact that cross-lingual links are possible between English and Amharic directly as well as between English or Amharic and semantics means that the depth of translation is flexible in L^3.

Shallow (or transfer) vs. deep (or interlingua) translation is a distinction going back to the early days of RBMT. Deep translation (e.g., Bond et al., 2011; Mel'čuk and Wanner, 2006) makes use of a language-independent semantic level, rules mapping source-language lexical items and structures to semantic units, and rules mapping semantic units to target-language lexical items and structures. The main advantage is that it is possible to translate from any of the system's source languages into any of its target languages without special-purpose rules for language pairs. The drawback of the deep approach is the difficulty faced in devising a semantics that is abstract enough to cover a large set of languages. The addition of new languages may necessitate changes in the semantics, which may in turn require changes in all of the source-to-semantics and semantics-to-target interfaces. This disadvantage is mitigated to some extent when translation is limited to narrow domains (Ranta et al., 2010).

Shallow approaches do not suffer from these problems; since the rules apply only to a specific language pair, there is no need to search for abstract general representations and no need to update the whole system when new language

[4] Since the Amharic nouns ሐኪም *hakim*'doctor' and በሽተኛ *bəštəña* 'patient' can be either masculine or feminine, the Amharic sentence shown is only one of four possible translations for the English sentence.

pairs are added. In addition, shallow MT is normally more efficient than deep MT because less processing is required at both ends.

Given the advantages of both approaches, it would make sense to integrate them in some way. Human translators seem to use such an eclectic approach, relying on a deep understanding when they need to, but "faking it" and making use of shortcuts when they can or when they lack the knowledge required for a deep understanding (Byrne, 2006). As far as we know, however, all existing MT systems operate either one way or the other.

L^3 integrates both shallow and deep approaches into a single system. It is straightforward to simply tell L^3 to make use of source-to-semantics and semantics-to-target links or only source-to-target cross-lingual links during translation. When they are available, the latter will always be faster, involving fewer variables and fewer constraints to satisfy. For example, consider the translation pair illustrated in Figure 2. For translation of the English sentence into Amharic, 3544 constraints are required for the deep approach, while only 2879 are required for the shallow approach. For translation of the Amharic sentence into English, 5594 constraints are required for the deep approach, 4371 for the shallow. There is a savings in the time required for constraint satisfaction of 28.1% in the former case, 30.3% in the latter.

3.4. Node mismatch

For cases where words in the input sentence do not correspond to explicit semantic units, XDG makes use of del arcs to implicitly delete the nodes on the SEM dimension. When there is a node mismatch in the opposite direction, however, this solution will not work (Pelizzoni and Nunes, 2005). This happens, for example, in generation, when the semantic input has no nodes for words that must appear in the output, such as determiners and auxiliary verbs. It also happens frequently in translation. For example, like many other languages, Amharic is a zero-subject ("pro-drop") language which may have no explicit subject. Like most other Semitic languages and many Bantu languages, it is also a zero-object language which may have no explicit direct or indirect object when this is coded as an affix on the verb. Thus in the English translation of an Amharic sentence such as አዳናት *adanat* 'he cured her', consisting of a verb only, the nodes for the English pronouns *he* and *her* must come from somewhere.

For this purpose, L^3 includes the possibility of **empty nodes**. Empty nodes are created during processing on the basis of **trigger nodes** associated with explicit input words. There are several types of empty nodes for different syntactic contexts. Here we focus on empty nodes for verb arguments that may not be explicit in the source languages. In the lexicon for Amharic and other such languages, finite verb entries include an attribute for a subject empty node, and finite transitive verbs include an attribute for an object empty node. Each of these empty node categories in turn has its own lexical entry (indicated by @SBJ and @OBJ in what follows), specifying a set of constraint attributes just as for any other entry. Thus verb nodes are the triggers for these argument empty nodes. When an input Amharic sentence contains a finite verb, a subject empty node is automatically added to the node list and assigned to the empty subject lexical entry. If the verb is transitive, an object empty node is also created and assigned to the empty object lexical entry.

The key property of the subject and object empty nodes is that there is no way of knowing prior to constraint satisfaction whether they will be needed or not. If the input Amharic sentence has no explicit subject (or object), the empty nodes get realized as explicit nodes in the English output:

አዳናት ⇒ አዳናት @SBJ @OBJ
⇒ {*cured*, *he*, *her*}
⇒ *he cured her*

If, on the other hand, the input Amharic sentence has an explicit subject (or object), the nodes remain empty in the English output. In the following sentence, for example, the noun አስቴርን *astern* 'Esther (acc.)' plays the role of object:

አስቴርን አዳናት ⇒ አስቴርን አዳናት @SBJ @OBJ
⇒ {*Esther*, *cured*, *he*, *zero*}
⇒ *he cured Esther*

To handle these cases we have introduced a set of empty node constraints in XDG. Informally, the constraints specify that the verb can have only one daughter along a sbj (obj) arc. If there is an explicit subject (object), the node for this constituent's head plays the role of daughter along the sbj (obj) arc from the verb, and the corresponding empty node remains empty in the target language. If there is no explicit subject, the empty node plays the role of sbj (obj) daughter in Amharic. Because of Amharic agreement constraints, the empty node is constrained to have particular person, number, gender, and case features. This node is realized in English as an explicit pronoun. Which pronoun it becomes is determined by English agreement constraints: each candidate pronoun has features that must agree with the features of the empty node, and the selection of the appropriate pronoun is accomplished through constraint satisfaction.

3.5. Translation

Given the modifications of the basic XDG framework discussed above, translation proceeds more or less in the same fashion as parsing and generation in XDG. As an example, consider the translation of the sentence *the doctor cured the patient* into Amharic. If the input language is morphologically complex, it is first processed with a morphological analyzer, resulting in one or more combinations of lemmas and grammatical features for each node. This is not the case for our English input sentence, however. Next lexicalization searches for entries in the source language lexicon that match the input words or lemmas. In addition to language-specific attributes, such as valency, agreement, and order, the matched lexical entries may also provide cross-lingual links, either to semantics or to the target language. These links are traversed during lexicalization, and the attributes in the "language" on the other end of the link are copied in the relevant node entry. For example, via cross-lingual links, node 3, corresponding to *cured* in the input, gets features from the lexical entry for the Amharic translation of *cure*, አዳነ *adanə*, including the general valency, order, and

agreement constraints of Amharic transitive verbs.

Finally, constraint satisfaction applies, as before. For a morphologically complex target language, as in this example, the result is a set of ordered target-language lemmas along with grammatical features. The final step is morphological generation, which yields target-language surface forms. We use our in-house Amharic morphological generator for this purpose (Gasser, 2011a).

As noted already for the monolingual case with respect to parsing vs. generation, the fact that all of the information in the lexica is declarative means that the same knowledge can be used for translation in both directions. An exception is the case of empty nodes, which are deleted in one direction and inserted in the other. Thus, given separate general mechanisms for handling empty nodes in the two directions, the fact that the system has the knowledge to allow it to translate the English sentence *the doctor cured the patient* into the Amharic sentence ሐኪሙ በሽተኛውን አዳነው *hakimu bəštəñawn adanəw* (among other possibilities) means that it can translate the Amharic sentence to the English sentence.

3.6. Structure mapping

An important concern in the design of MT systems is that they have the capacity to represent the structural divergences between languages. Dorr (1994) provides one classification of the sorts of divergences that can occur.

Here we discuss an English-Amharic example illustrating two of Dorr's divergence types: thematic and categorial. Amharic has a set of impersonal experiential verbs whose subject agreement is always third person singular masculine and whose object suffixes agree with the experiencer argument. If the experiencer also takes the form of an explicit argument of the verb, this argument is unmarked for case, like a subject, but it agrees with the verb's object suffix rather than with the verb's subject affixes. We refer to this argument as a "topic". An example is the verb ደከመ *dəkəmə* 'be tired': ደከማት *dəkəmat* 'she is tired', lit. 'it tired her'; አስቴር ደከማት *aster dəkəmat* 'Esther is tired', lit. 'Esther, it tired her'. For these verbs Amharic and English differ in two ways. English uses an adjective, along with a form of *be*, while Amharic uses a dedicated verb. In English the subject of *be* is the experiencer, while in Amharic the experiencer is cross-indexed as obligatory object agreement on the verb and (if present) as the syntactic topic of the sentence.

In L^3 structural divergences are handled on interface dimensions. Consider first the lexical entry for the Amharic verb ደከመ *dəkəmə* 'be tired'. With respect to the Amharic ID and LP dimensions, this is a transitive verb with the unusual property that it may have a topic argument (the experiencer) that agrees with the verb's object morphology. Because the topic may not be realized and because it corresponds to an explicit argument in the semantics and in languages such as English, the verb acts as a trigger for a **topic empty node**.

In Entry 3, we show some of the information in the English and Amharic lexica that enables translation of these sentences. For Amharic we show a portion of the entry for the verb lemma ደከመ *dəkəmə*. This acts as a trigger for an empty topic node (with lexical entry @TOP), which requires an outgoing arc in the ID dimension with a top label. Also on the ID dimension, there is an agreement constraint attribute specifying that the topic must agree with the object suffix on the verb on the person, number, and gender (png) feature. The Amharic verb has a cross-lingual link to semantics that associates this lemma with the semantic lemma TIRED and stipulates that the arg1 arc from this node on the SEM dimension should go to the node that has an incoming top arc on the Amharic ID dimension.

For English two relevant entries are shown. The first is for the verb *be* with predicate adjectives (as in *be tired*). This entry stipulates that this node must have an outgoing arc in the ID dimension with a padj (predicate adjective) label. This entry also has a cross-lingual link to semantics that deletes the associated node on the SEM dimension. Finally, we show a portion of the lexical entry for the English adjective *tired*. It stipulates that this word must have an incoming arc in the ID dimension with a padj label. There is also a cross-lingual link to semantics that associated this word with the semantic lemma TIRED and stipulates that the arg1 arc from this node on the SEM dimension should go to the node that has an incoming sbj arc on the English ID dimension.

Figure 4 illustrates the translation of the sentence *she is tired* into Amharic. L^3 returns two translations in this case, one with and one without the explicit topic pronoun እሷ *ɨswa* 'she' (indicated in the figure by the gray circle for node 1 in Amharic ID). The same entries shown in Entry 3 would enable translation in the reverse direction.

Entry 3 TIRED in English and Amharic

```
AMHARIC
- lemma: dekeme
  empty: [@TOP]
  ID:
    out: {top: !}
    agree: [[top, obj, png]]
  cross:
    sem:
      lex: TIRED
      IDSem:
        linkend: {arg1: [top]}
ENGLISH
- lemma: be_padj
  ID:
    out: {padj: !}
  cross:
    sem:
      lex: zero
- lemma: tired
  ID:
    in: {padj: !}
  cross:
    sem:
      lex: TIRED
      IDSem:
        linkend: {arg1: [sbj]}
```

4. Project status and ongoing work

Because we are making some basic modifications to XDG, we have re-implemented the framework from the bottom up. Our implementation is in Python; lexica are encoded in YAML format. All of our software, includ-

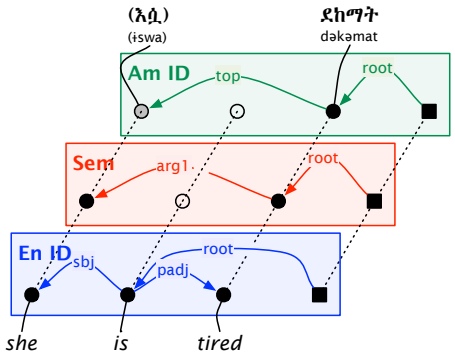

Figure 4: Translation of *she is tired* into Amharic.

ing the XDG implementation, lexica, and morphological analyzer and generator for Amharic, is available under a GNU GPL3 license at `http://www.cs.indiana.edu/~gasser/Research/software.html`.

It is premature to attempt an evaluation of our very rudimentary English-Amharic translation system.[5] Our next step is to augment both the English and Amharic grammars with more structures and to apply it to translation within the restricted domain of arithmetic and to computer-assisted translation within the domain of economics. In parallel with this work, we are developing an API for the grammar framework that will enable users to create simple grammars for additional languages. We are also exploring ways to integrate machine learning into the framework by using existing Amharic-English parallel corpora to aid in lexical disambiguation.

5. Conclusion

In this paper, we have introduced L^3, an evolving framework for RBMT and RBCAT that we are applying to the development of an English-Amharic MT system. We have discussed and illustrated some of the features of L^3: its bidirectionality, its capacity to handle structural divergences between typologically diverse languages such as English and Amharic, and its integration of shallow and deep translation into a single system. Although our Amharic-English MT system is still only a toy, by focusing on features that distinguish the languages, we feel we are on the right track.

References

Barreiro, A., Scott, B., Kasper, W., and Kiefer, B. (2011). Openlogos machine translation: Philosophy, model, resources, and customization. *Machine Translation*, 25(2):107–126.

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

Bond, F., Oepen, S., Nichols, E., Flickinger, D., Velldal, E., and Haugereid, P. (2011). Deep open-source machine translation. *Machine Translation*, 25(2):87–105.

Byrne, J. (2006). *Technical Translation: Usability Strategies for Translating Technical Documentation*. Springer, Dordrecht, the Netherlands.

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.

Dorr, B. (1994). Machine translation divergences: a formal description and proposed solution. *Computational Linguistics*, 20(4):597–633.

Forcada, M. L., Ginestí-Rosell, M., Nordfalk, J., O'Regan, J., Ortiz-Rojas, S., Pérez-Ortiz, J. A., Sánchez-Martínez, F., Ramírez-Sánchez, G., and Tyers, F. M. (2011). Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation*, 25(2):127–144.

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.

Gasser, M. (2011a). HornMorpho: a system for morphological processing of Amharic, Oromo, and Tigrinya. In *Proceedings of the Conference on Human Language Technology for Development*, Alexandria, Egypt.

Gasser, M. (2011b). Towards synchronous extensible dependency grammar. In *Proceedings of the Second International Workshop on Free/Open-Source Rule-Based Machine Translation*, Barcelona.

Mayor, A., Alegria, I., de Ilarraza, A. D., Labaka, G., Lersundi, M., and Sarasola, K. (2011). Matxin, an open-source rule-based machine translation system for Basque. *Machine Translation*, 25:53–82.

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

Paolillo, J. (2005). Language diversity on the internet: Examining linguistic bias. In UNESCO Institute for Statistics, editor, *Measuring Linguistic Diversity on the Internet*. UIS, Montreal, Quebec, Canada.

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.

[5] Evaluation of our Amharic morphological analyzer and generator has been reported on elsewhere (Gasser, 2011a).