K. Károly, Á. Fóris (eds.) *New Trends in Translation Studies (In Honour of Kinga Klaudy).* Budapest: Akadémiai Kiadó, 2005. pp. 207-218.

Machine Translation and the Rule-to-Rule Hypothesis

Gábor Prószéky
MorphoLogic
proszeky@morphologic.hu

Keywords: machine translation, pattern-based, rule-to-rule hypothesis, English, Hungarian.

Abstract

The rule-to-rule hypothesis says that every syntax rule has its counterpart in semantics. If we replace semantics with translation, we get a basic concept a machine translation system can rely on. Syntax-translation pairs are represented by pairs of patterns where pattern can stand for both rules and lexical items. Combining the advantages of example-based and rule-based machine translation, a new paradigm, pattern-based translation is introduced. The system called MetaMorpho based on these principles has been tested for English-Hungarian translation, and showed very promising results both in translation quality and speed.

1. Rule-to-rule Translation

The meaning of a complex linguistic structure is wholly determined by its sub-structures and the meanings of them. In the Rosetta machine translation system (Landsbergen 1985) we can meet a rather direct application of the compositionality principle: *„The meaning of an expression is a function of the meaning of its parts and the way in which they are syntactically combined. This principle was adopted from Montague Grammar (Thomason 1974). Obviously, this principle will lead to an organization of the syntax that is strongly influenced by semantic considerations. But as it is an important criterion of a correct translation that it is meaning-preserving, this seems to be a useful guideline in machine translation."* (Appelo et al 1987) Semantic compositionality was formalized by the rule-to-rule hypothesis of Bach (1976): it says that a tight correspondence is imposed between syntax and semantics such that every rule of syntax is also a rule of semantics.

In the reality, the meaning of a complex structure cannot always be built by a function of its parts, therefore it should be treated as an unstructured unit. This is how structures are described in construction grammar (Goldberg 1995). Perfect harmony between syntactic and semantic composition rules can only be found in artificial languages. The rule-to-rule hypothesis seems, however, a useful working assumption in machine translation as well: if a structure can be described syntactically in the source language, it can also be described by structures of the target language. Of course, human languages are not as exactly formalized as formal languages. In cases of ambiguous source language sentences, more than one target language description is permitted to produce. If a source structure is underspecified, target structures are minimally as underspecified as the source was. Technically, the only thing that can be introduced is an extra layer to a phrase structure (sub-)tree, what we call the translation of the actual sub-tree. So a compositional approach to translation will have a representation of the contribution of each word and sub-phrase towards the translation of the whole.

2. Rule-based and Example-based Machine Translation Strategies

As it is well-known, three well-known different rule-based approaches to MT are traditionally distinguished: direct, interlingual and transfer. The direct method is the strategy adopted by most early MT systems. It uses a primitive one-stage process in which words in the source language are replaced with words in the target language and then some rearrangement is done. The main idea behind the interlingua method is that the analysis of any source language should result in a language-independent representation. The target language is then generated from that language-neutral representation. The transfer method first parses the sentence of the source language. It then applies rules that map the grammatical segments of the source sentence to a representation in the target language. The three methods are usually shown with the help of source language–interlingua–target language triangles (Figure 1).

Figure 1. The basic machine translation strategies

On the other hand, example-base machine translation (EBMT) was suggested in the eighties, for example, by Nagao (1984) as a better approach to machine translation than rule-based machine translation (RBMT). Since then, several authors, like McTait (2001) and Carl (2001) have pointed out that the performance of EBMT can considerably be improved by adding linguistic background knowledge to the system. The goal of our research was to find an optimum between EBMT and RBMT in terms of practical applicability: translation quality and speed. EBMT is generally considered a statistics-based, probabilistic process, whereas RBMT is often thought of as a fixed, traditional, deterministic approach. In contrast, we believe that EBMT and RBMT are just the two extremes of a generalized model. In our model, there is an arbitrary number of possible transitions between the two. Not all of our "examples" have been directly extracted from corpora, or produced by statistical analysis. Rather we opted to build a database of structural segments, which have been generated from various sources: not only corpora, but various dictionaries of idioms and collocations. Besides multi-word lexemes we had to add some single-element lexical entries to the collection, as well. However, in many cases, their linguistic behavior has been described by expanding them to multi-word units. Namely, words do not occur alone, but in context: we have added context to them: instead of a single entry for the English verb *add* we have made *add a nought* ('hozzáír'), *add a piece to [N]* ('megnagyobbít'), *add [NUM] to [NUM]* ('összead'), *add to [N+GEN] difficulties* ('növeli vk nehézségeit'), *add to [N+building]* ('hozzáépít'), etc.

3. Pattern-based Language Description

In our MetaMorpho formalism both what are traditionally called rewriting rules and lexical entries are integrated in the form of patterns. Currently we have more than 200 thousand patterns, the majority of which are lexicalized items. The system uses no separate dictionary: what would traditionally be lexical entry is integrated in the form of patterns. A lexical and a grammatical pattern are shown by Example 1.

```
NX[N.lex, N.num] = N(lex="dog")

VP[TV.conj] = TV(lex="meet", pass=NO) + DOBJ
```

Example 1.

The patterns in the grammar are much more complex than the ones shown. For the sake of human readability and maintainability, lexical items are coded in a simpler form where all non-lexical information is omitted. The actual rules are then generated off-line from their simplified source. A large amount of linguistic knowledge is effectively encoded in this conversion. The philosophy behind this is to remove the burden of interpreting a complex and linguistically motivated formalism from the parser while representing the same linguistic knowledge in an off-line step.

The grammar itself operates with pairs of patterns that consist of one source pattern used during bottom-up parsing and one or more target patterns that are applied during top-down generation. A pair of patterns looks like the ones in Example 2: they describe the basic use of English words *dog* and *meet (somebody)* and their Hungarian translations.

```
(1) *NX=dog:0401311411
(2) EN.NX[N.lex, N.num] = N(lex=”dog”)
(3) HU.NX = N[lex=”kutya”, NX.case, NX.ownernum, NX.ownerpers]

(4) *VP=meet+DOBJ:0401311343
(5) EN.VP[TV.conj] = TV(lex=”meet”,pass=NO) + DOBJ
(6) HU.VP(focus=NO,EN.DOBJ.reqfocus=YES) = DOBJ[case=INS] + TV[lex=”találkozik”,VP.tense]
(7) HU.VP = TV[lex=”találkozik”, VP.tense] + DOBJ[case=INS]
```

Example 2.

The lines show the name of the grammatical phenomenon described by the patterns (1,4), the English source pattern (2,5), and the Hungarian target patterns (3, 6 and 7). (6) is used when direct object of *meet* requires focus position in the Hungarian output, and (7) is used when it does not.

Every terminal and non-terminal symbol (or what is equivalent, the corresponding node in the syntactic tree under construction) has a well-defined set of features. The number of features varies between zero and a few dozen, depending on the category. These features can either take their values from a finite set of symbolic items (e.g., values of `case` can be `INS, ACC, DAT,`

etc.), or represent a string (e.g., `lex="meet"`, that is, the lexical form of a token). The formalism does not allow for embedded feature structures. It is important to note that no structural, semantic or lexical information is amassed in the features of symbols: the interpretation of the input is contained in the syntactic tree itself, and not in the features of the node on the topmost level.

A pattern can be "productive," which means it contains little or no lexical information. Such a pattern would be, for instance, `VP=TV(vti=VT)+DOBJ`, which describes the fact that a transitive verb and an object can form a verbal phrase. Such patterns are traditionally called rules. Partly or fully lexicalized patterns, on the other hand, contain rather specific lexical information, e.g. `VP=TV(lex="meet")+DOBJ, VP=TV(lex="count")+PPOBJ(lex ="on")`, expressing that *meet* needs a direct object, or *count* has a prepositional object with *on* as its valence.

```
English input:
  1---2----3---4----5-----6--7-------8
  Jim does not sink money in anything.
  1--
  Jim
      2----
      tesz
           3---
           nem
               4------
               mosogató
               süllyed
               süllyeszt
      2----3---4----
      nem süllyed
      nem süllyeszt
                    5-----
                    pénz
                             7--------
                             bármi
                    5-----6--7--------
                    pénz bármiben
               4----5-----6--7--------
               befektet pénzt bármibe
               4----5----
               pénzt süllyeszt el
               befektet pénzt
      2----3---4----5-----
      nem fektet be pénzt
      2----3---4----5-----6--7------
      nem fektet be pénzt semmibe
  1---2----3---4----5-----6--7------8
  Jim nem fektet be pénzt semmibe.
```

Example 3.

More specific patterns (e.g. *count on*) can override more general ones (e.g. *count*), meaning that all subtrees containing symbols that were created by the general pattern are deleted. Every symbol that is created and is not eliminated by an overriding pattern is retained even if it does not form part of a correct sentence's syntactic tree. Each pattern can state any number of overrides on other rules: if the overriding rule fires over a specific range of the input, it blocks the overwritten one over the same range. *Example 3* shows some of the sub-structures and an overriding made while parsing.

Not only is this extremely useful for debugging purposes, but it allows for "best guesses" when no interpretation for a whole sentence is found in real-life applications.

4. Applying Rule-to-Rule Translation

The analysis of the input is performed in three basic steps. First the sentence to be translated is segmented into terminal symbols or tokens. This token sequence is the actual input of the parser. The morphological analyzer determines all the needed morpho-syntactic attributes of these symbols. We use the Humor analyzer (Prószéky & Kis 1999) that is based on surface patterns. The basic strategy of Humor is inherently suited to parallel execution: search in the main dictionary, secondary dictionaries and affix dictionaries can be performed in a parallel way. In case of agglutinative languages like Hungarian, where the number of inflected word-forms for a single word is well over hundreds, a reliable morphological generator is a crucial part of any translation tool. The advantage of Humor is that it can be used as a generator as well as an analyzer. The system accepts unknown elements: they are treated as strings to be inflected at the target side. In fact, not the string, but its "pronounced" transcription is inflected by the Hungarian generator (Example 4).

```
I met Ms. Gerber.
Találkoztam Ms. Gerberrel.

I met Mr. Isabelle.
Találkoztam Mr. Isabelle-lel.

I met Mrs. Bordeaux.
Találkoztam Mrs. Bordeaux-val.
```

Example 4.

Second, Moose, a bottom-up parser (Prószéky et al. 2004) analyzes this input sequence and if it is recognized as a correct sentence, comes up with one or more root symbols. When the whole input is processed and no applicable patterns remain, the target equivalent is read top-down from the root symbols by firing the target pattern corresponding to the source pattern that created the edge at parse time. This solution we can call “immediate transfer” as it uses no separate transfer steps or target transformations.

Pattern pairs can have conditions in the left-hand side, and in the case of multiple target patterns, the first one whose conditions are satisfied is fired. The right-hand side of the source pattern can state conditions for any of its symbols’ values. To handle more complicated word-order changes, however, a stronger means of rearrangement is also provided: interpretation of the source structure in the target structure may need rearrangement of its elements within the scope of a single node and its children. Three subtree interpretations are allowed: (i) permutation of the node’s children, (ii) deletion of one or more children from the target tree, and (iii) insertion of terminal symbols. Example 5 shows MetaMorpho’s translation trees for the sentence *I have gone home*. Its translation is *Hazamentem*. Numbers in curly brackets show the number of the source pattern belonging to the Hungarian translation.

```
EN.S_ROOT 2457
    S_FULL 2454
    |---CS 1609
    |   |---SUBJ 64
    |   |   |---NP 59
    |   |       |---PRON lex="I", num=SG, pers=P1, case=NOM
    |   |
    |   |---MPRED 1601
    |       |---V lex="have", form=F1
    |       |---PRED 1600
    |           |---VP 1561
    |               |---TV 245
    |               |   |---V lex="go", form=F3
    |               |
    |               |---PART lex="home"
    |
    |---PUNCT lex="period"

HU.S_ROOT 2460{2457}
    S_FULL 2461{2454}
    |
    |---CS 2462{1609}
    |   |---SUBJ 2471{64}
    |   |   |---PRON num=SG, pers=P1, case=NOM
    |   |
    |   |---PRED 2472{1600}
    |       |---V ik="haza", lexb="megy", num=SG, pers=P1
    |
```

```
|---PUNCT lex="period"
```

Example 5.

There are various differences in the source and the target structure: the Hungarian tense system essentially simpler than English, but compounding is very productive and non-third person subject of sentences are not explicitly given in most cases. Thus, *I* is translated as a verbal suffix, the present perfect structure expressed by *have* plus the verb's third form becomes simple past in Hungarian and *go home* is expressed by a single word in Hungarian: *hazamegy*. MetaMorpho's full parsing of this English sentence needed 2458 steps, the synthesis of the Hungarian output was made in 26 steps. This big difference between the numbers of steps in analysis and generation is of general importance: it illustrates that the output is perfectly given when the parse is over, so the only operation to be done is simplification of the target description of the root element of the analysis.

A subtree can be memorized in a feature when a unification operation takes places at parse time, and because this feature's value can percolate up the parse-tree and down the target tree, just like any other feature, a phrase swallowed at any level in the source side can be expanded at a completely different location in the target tree. The power and simplicity of subtree memorization and random insertion can be demonstrated in Example 6 with the translation of English possessive structures into Hungarian: *the eighteenth birthday of Kinga* translates into *0*-DET/*the Kinga*-N/*Kinga tizennyolcadik*-NUM/*eighteenth születésnapja*-N+POS/*birthday*, that is, the order of the two nominal phrases is reversed and the determiner absorbed by the proper noun: *Kinga tizennyolcadik születésnapja*.

```
EN.NP_ROOT 244
    DET lex="the"
    NM 234
        NNY 134
            ADJP 118
                NUM lex="eighteen"
            NZ 131
                N lex="birthday", num=SG
        PREP lex="of"
        NP 225
            N lex="Kinga", num=SG

HU.NP_ROOT 320{244}
    NP 323{225}
        N lex="Kinga", num=SG, case=NOM, postp=""
    NM 324{234}
        ADJP 338{118}
            NUM lex="tizennyolc"
```

```
NZ 339{131}
    N lex="születésnap", num=SG, case=NOM, postp=""
```

Example 6.

Through the interplay of only two patterns (the place of memorization at N-bar level and insertion at NP-level), a possessive structure of any length is translated recursively in reverse order into Hungarian.

As a consequence, we can say that MetaMorpho's translation method is opposed to traditional transfer approach in that there is no need to "transfer" an abstract structure at any level: we create our analysis with the final output in mind, and can produce the result in a very straightforward manner, without any need for complex independent transfer methods following syntactic analysis. However, MetaMorpho does not use interlingual representations, thus it would be misleading to claim that it belongs to direct translation systems. Unlike the first primitive machine translation systems, MetaMorpho uses systematic grammatical descriptions and a mechanism that is a variant of the rule-to-rule idea of Bach: the target equivalent of a MetaMorpho source structure is its translation and not the formal logical representation of its semantics. Summarizing the above, MetaMorpho seems to belong to a fourth machine translation paradigm. Its shows, of course, some relation to the Rosetta machine translation system (Landsbergen 1984) which uses logical semantic representations. Rosetta really used the rule-to-rule hypothesis, but that representation was considered as an interlingua what differs basically from the MetaMorpho approach. Therefore, we introduce a fourth approach to machine translation for MetaMorpho, as it is represented by Figure 2.

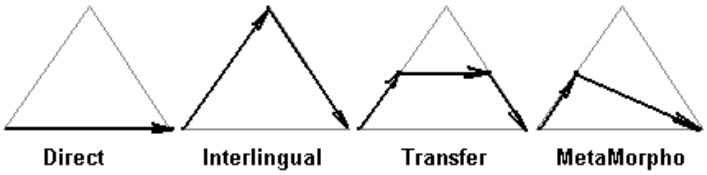

Figure 2: Strategies of MT solutions with MetaMorpho

5. Implementation Details

The principles and modules discussed above are used in a real-life application translating from English into Hungarian. The system, MetaMorpho, is designed to provide a translation quality that should be high enough to serve both comprehension assistance and authoring needs. Its modules have been written in programming language C^{++}. The basic linguistic modules, core patterns have been written and tested in a special development environment. The number of these core type patterns for a language is surprisingly low: it is in the magnitude of 1000 for a language like English. These basic patterns serve as generalized examples for the more specific ones. Lexical patterns have mainly been derived from existing lexicons and collocation databases.

The motivation for creating Moose, the robust bottom-up parser (Prószéky et al. 2004) is that the grammar's applications invariably require access to a parse's partial results in the absence of a full parse tree. The parser invokes a user-defined filter when parsing is complete. These filters have access to all parse trees and can select, for instance, a disjunctive coverage of the input tokens.

Many machine translation systems described as promising in the literature could not reach their full potential, mainly because it was very difficult to expand their lexicon to a size that can be used without problems by an average user. To avoid this, we have also implemented a grammar writer's workbench, called RuleBuilder. This allows the controlled addition of new, lexical or even syntactic patterns into the grammar. With the help of this, the (authenticated) user can add and modify the rules of the grammar on-line without the need to recompile the rest. Around this interface a grammar development workbench with many debugging features has been built to facilitate the work of grammar writers.

We have to note, nonetheless, that even though our coders found the strict descriptive formalism acceptable and had extensive sample sets to work with, it required a great effort to coordinate their work and obtain coherent results. At any rate, coding was followed by thorough manual testing. As opposed to the rather low number of core patterns, the number of lexical patterns is well in the hundreds of thousands. Assuming that one pattern can be stored in 100 bytes and a typical PC can be expected to have 100 megabytes of free RAM, the maximum number of patterns that might be used by the system can even be around the magnitude of million.

6. Future Work

We have started to work on a combination of the MetaMorpho machine translation tool and translation memories. The integration with an SQL database forms the basis of this combined solution (Hodász et al. 2004). Taking advantage of MetaMorpho's ability to translate incomplete sentences, we could translate this differing part of the sentence and thus improve the efficiency of translation memories. MetaMorpho currently fetches only the first target equivalent from the lexical patterns. This could be changed by reordering the target equivalents according the context. A word-sense disambiguation module providing semantically disambiguated output is under development. We are also working on a topic recognition module running the same way as language identifier programs do but identifying the sublanguage (business, medicine, sport, etc.) having a well recognizable terminology within a single language, say English. Once the topic identifier determines the topic, it sets MetaMorpho's lexical patterns' target equivalent ranking accordingly.

7. Summary

MetaMorpho is an innovative system in many ways. It applies the rule-to-rule hypothesis of Bach for translation purposes. The system relies on a uniform description of lexical and structural information called patterns: they are basic tools for describing both standard and idiomatic behavior of sentences, clauses and phrases. If a pattern is short and fully specified, it is a lexical entry in the traditional terminology. If it is longer, but fully specified, it is an idiom, or a specific example. If no attributes of a pattern are specified, then the pattern is conventionally a rule. Our approach puts the emphasis on the transitions between the two: idioms and collocations are elements that are filled in, but which are not fully specified. The key issues of our model are how to manage these generalized patterns. MetaMorpho patterns show certain similarities to the "translationally equivalent patterns" used in the English-Japanese translation system of Kawasaki et al. (1992). The knowledge base in their model consists of patterns mainly utilized to translate idiomatic or nonstandard expressions.

The main reason why the pattern-based idea has not been generally applied is memory limits of the earlier computers. MetaMorpho represents a generalization of the EBMT model, however, parsing is not statistical, and it combines source language analysis and target language interpretation in one single task. If the input is grammatically correct, the system should provide correct translation, and if the input cannot be analyzed, the system should provide the translation of all the separate correct structures it can identify.

MetaMorpho does not use interlingual representations. Its approach, however, uses no transfer steps after parsing, because both structural and lexical transfer have already been done while parsing. Thus, it would be very misleading to say that our approach belongs to the paradigm of direct translation, just because it is neither interlingual, nor transfer-based. Unlike the first primitive machine translation systems, MetaMorpho uses systematic grammatical descriptions and a mechanism that is close to the rule-to-rule hypothesis of Bach. The target equivalent of source structure is its translation and not the formal logical representation of its semantics. MetaMorpho seems to belong to another machine translation paradigm, even it shows some relation to machine translation systems which use logical semantic representations (e.g. Rosetta). The original form of the rule-to-rule hypothesis in those systems was, however, used as an interlingua which essentially differs from the MetaMorpho approach.

8. References

Appelo, L., C. Fellinger, J. Landsbergen. 1987. Subgrammars, Rule Classes and Control in the Rosetta Translation System. In: B. Maegaard (ed.) *Proceedings of the Third Conference of the European Chapter of the Association for Computational Linguistics*. Copenhagen, 118-133.

Bach, E. 1976. An extension of classical transformational grammar. *Mimeo*. Amherst: University of Massachusetts.

Carl, M. 2001. Inducing Translation Grammars from Bracketed Alignments. In: B. Maegaard (ed.) *Proceedings of the Workshop on Example-Based Machine Translation.* Santiago de Compostela: [http://www.eamt.org/summitVIII/ workshop-papers.html]

Goldberg, A.E. 1995. *Constructions: A Construction Grammar Approach to Argument Structure.* Chicago: The University of Chicago Press.

Hodász, G., B. Kis, T. Gröbler. 2004. Translation memory as a robust example-based translation system. In: J. Hutchins (ed.) *Proceedings of the 9th EAMT Conference*. La Valletta: Foundation for International Studies. 82-89.

Kawasaki, Z, F. Yamano, N. Yamasaki. 1992. Translator Knowledge Base for Machine Translation Systems. *Machine Translation* 6(4), 265-278

Landsbergen, J. 1984. Isomorphic Grammars and their use in the Rosetta Translation system. In: King, M. (ed.) *Machine Translation Today.* Edinburgh: University Press, 351-372.

McTait, K. 2001. Linguistic Knowledge and Complexity in an EBMT System Based on Translation Patterns. In: B. Maegaard (ed.) *Proceedings of the Workshop on Example-Based Machine Translation.* Santiago de Compostela: [http://www.eamt.org/summitVIII/ workshop-papers.html]

Nagao, M. 1984. A Framework of a Mechanical Translation between Japanese and English by Analogy Principle. In: A. Elithorn and R. Banerji (eds.): *Artificial and Human Intelligence*, Amsterdam: North Holland. 173-180

Prószéky, G., B. Kis. 1999. Agglutinative and Other (Highly) Inflectional Languages. In: R. Dale, K. Church (eds.) *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics.* College Park, Maryland. 261–268.

Prószéky, G., L. Tihanyi, G. Ugray. 2004. Moose: a robust high-performance parser and generator. In: J. Hutchins (ed.) *Proceedings of the 9th EAMT Conference*. La Valletta: Foundation for International Studies. 138-142.

Thomason, R.H. (ed.) 1974. *Formal Philosophy: Selected Papers of Richard Montague.* New Haven: Yale Univ. Press.