

Overview of the system Sept. 2004 – notes

Søren Harder

October 1, 2004

Abstract

These are notes to a description of my translation system. It is meant to give an overview and function as an 'accumulation container' for thoughts about problems and solutions, to-do's, 'what the system can(not) do and why (not)'. The general tone of this document is often negative, as it is not meant as a description of the design principles, but of the problems of implementing these. If and when the following notes are included in the main thesis, they will be written in a more 'positive' tone.

1 Depsplicator

The 'dependency explicator', or depsplicator is explained in the (very incomplete) document 'Explication of dependency structures' (Dec. 2003). I will in the following give a much shorter, but more complete overview of it.

Input The system as a whole takes as input sentences analysed by the Danish VISL Constraint Grammar. The format on which the system has been developed and tested is the format of treebank-files, such as `citat90a_0-AHOVEREB` and `citat90first`, which I call `tb1` and `tb2-tb5`. The system also exist in a 'live' version on the VISL homepage, where it is running, but not thoroughly tested. Here it is using the format of the 'live' CG-analyses.

Output from the 'Depsplicator' In the stand-alone version this module gives output in a series of formats (see 'Explication of dependency structures'). The conceptually most simple is the CG input plus three additional tags:

- An index for the word: each word is assigned a number (1, 2, 3, ...). This tag is marked by '='.
- The index of the head: the index of the word, which the word is dependent on. Marked by ':'.
- The relative index of the head. This information is redundant and calculated by subtracting the two other. It tells how far the head is to the right (marked by '+') or left (marked by '+-').

The output that is passed on to the following modules is the internal structure: a Prolog-term, with the predicate 'word', with 9 arguments for the different types of information in the CG-analysis. The sixth is the dependency structure, which is a translation of the syntactic tags (the '@'-tags) into a five place Prolog-term, with the predicate 'dep', depicting the information from the CG about:

1. Function (the part of the '@'-tag that the arrow points away from)
2. Head (what the arrow points towards)
3. Index (not part of the CG-tag, but the information that the Depsplicator has been built to produce).
4. Direction (the (double) arrow of the tag or '*' if none).
5. Prefix (material before a hyphen: FS, ICL, AS, S or F)

In connection with the Depsplicator I have developed a suite of functions to extract the different types of information from this structure, and I use these functions in the rest of the system too.

Structure of algorithm Originally I believed the task of explicating the dependency could be solved as a 'classical Prolog problem' with no process-control. By this I mean, to define the meaning of the tags (e.g. 'A @SUBJ is dependent on a V') and leave the rest mostly to Prologs inbuilt backtracking. This showed not to be the case, so some extra modularity was introduced, splitting the task and introducing 'expert' modules. The depsplicator thus goes through these processes:

- Translation. CG-format to Prolog `word/9` structures.
- Conjunction. Conjunctions are searched for, conjoints localized and the dependency-relation of conjoints are unified.¹
- PSR. This is an abbreviation for 'phrase structure rules', but the component is more like a chunker, passing a window over the sentence to establish the dependency structure for unambiguous local structures as NPs etc. Contains 23 rules. Possibly the whole explication could have been built with this as its core.
- `sub_group/2`. Establishes a skeleton-structure of super- and subordinate clauses, by establishing left-edges/subordinators for each subordinate clause. This is used later to make sure, that constituents in subordinate clauses are not mistaken for constituents in superordinate clauses and vice versa.

¹This system is very complex and not very robust, especially the handling of more than binary conjunction poses problems. When I used my system in connection with my work at Aminova this module had to be skipped.

-
- `bind_dep/2`. Produces for each word a list of candidate heads.
 - `rel_clean/3`. Cleans up the data of `bind_dep/2`, so that the dependency of the MV-head of a relative clauses is to something outside the clause.
 - `sort_dep/4`. The combinations of the candidate dependencies from `bind_dep/2` are tried out under restrictions such as non-cyclicity, argument-uniqueness etc.
 - `output/4`. The structure is piped on to a process that prints it out in the format asked for, or passes it on to the translation component.

2 History of the translation algorithm

One of the original intentions of the PhD-project was to combine the available VISL CG-system with established (theoretical) MT-techniques, in particular rule-based techniques, to show that the CG-analyses could give a substantial boost to 'state-of-the-art' systems.

Unfortunately the 'state-of-the-art' was very heterogeneous, consisting of rule-based, non-rule-based and hybrid systems, with huge differences between commercial and academic systems. I ended up using a system intended for teaching MT as my starting point (Trujillo 1999).² This system fulfills many of the original constraints, but, along the way, I have discovered some problems with using it, and with the way I have used it.

On the plus side, it is available in source code, relatively simple and well documented, as it is the code for a coursebook in MT. This means that everything could be changed to suit my needs, also because it was written in my choice programming language Prolog.

One drawback was its extremely small grammatical coverage, as it is only intended as an exemplary grammar. This means I needed to write my own English grammar. That I needed to write my own Danish grammar (for use in the exploratory stages before I started using the CG as analysis component) was of course something I could not have expected being different. But that there wasn't anything but an extremely small English grammar to model this on, has shown itself as a problem.

Since the system covers many different types of translation, it also covers the one I decided to go with. On the other hand this means the modularity and design of it is intended to enable all the different types and this is often not necessarily the ideal design when you only need to use one of the techniques in one direction, producing a lot of 'lazy code' and illogical design.

My use of the original code by Trujillo has gone through the following phases:

- Exploratory use of the original English-Spanish system in all its modes.
- Development of a Danish grammar in place of the Spanish (The Pilot Project, as described in 'Three strategies for Transfer Machine Translation'). This included small extension of the English grammar as well. I

²PRELIMINARY NOTE: PROPER REF

used this Danish grammar in both directions for transfer-systems, but have not tested it on Interlingua-based systems. Constructing a two-level Danish morphology in the style of the English was too large a task, so I wrote a simpler, paradigm-based, morphology with the same interface as the original.

- Extending the pilot project grammar to handle one 'real' Danish sentence taken from the 'tb1' corpus and afterwards developing a program that can produce the same improvement (mainly) automatically from the CG-analysis. This meant that all other direction/transfer pairs than DA-EN lexical transfer were disregarded.
- Extending this system to handle more syntactic phenomena by developing it on the sentences in the tb1 from the start and on.
- Parallel to the previous: Extracting an approximate 58000 word Danish to English lexicon from the VISL-resources and embedding it in the system. This lexicon has no ambiguities (all Danish words have one English translation) and the system is not yet capable of handling multiple possible translations.

3 Overview of the translation algorithm

Basic idea of the system The basic philosophy of the system is to use the CG-analysis of a sentence as a double resource. On the one hand it is an analysis of the sentence, as the analyses provided by the grammars/parsers of the pilot system. On the other hand it exemplifies instances of language structure from which grammars (such as the ones used in the pilot project to produce its analyses) can be extracted.

This philosophy is taken very literally: the CG-analyses are actually translated into the exact same format as the rule-based analyses and the rule-based grammars. One of the reasons for this is that the system was intended as a supplement to a rule-based system, so that the rule based system is capable of giving high-quality translations of the relatively small fragment of language that the implemented grammar and lexicon covers, and when it opts out the processing is taken over seamlessly by the much more robust system driven on the CG-analyses.

Three things are important to note regarding the extraction of grammar rules. Firstly: the grammar is intended to be as lexical driven as possible, and it is the lexical items that are extracted and conjectured. The actual grammar rules (of which there are at present 25) are written by hand beforehand.

Secondly, even though the above description gives the air of a fully-fledged system for machine-learning of grammars, that runs through an analyzed corpus and outputs a grammar for the language, it is not intended to be that. The lexical items extracted will not necessarily be correct, but workable. There are no lexical typing and no abstraction from the concrete valency to a more abstract. The concept of valency and the distinction between valency licensed

arguments and non-licensed modifiers which is central to rule-based grammars, but hard to handle correctly in CG, will be further muddled.

With further work the system might become a system for machine-learning of grammars.³ This may be of great theoretical interest, but I am not sure it will cause any improvement in quality of translation, since this process is a process of generalization, which even though it results in a grammar, that describes its target, the whole of language, better than my conjectured lexical items describes their target, the single sentence under scrutiny, the information needed for translation is exactly information about the grammatical make-up of the sentence at hand.

The only way a machine-learned grammar may be an improvement on my 'ad-hoc' system, is by the fact, that the machine-learned grammar may be post-edited: it allows for errors to be weeded out and exceptions to grammar rules to be included, simply because the grammar is *there*. I believe, though, that the 'ad-hoc' system has ample possibility for fine-tuning, that will enable improvement of the system by continued development of the system by continuously testing it on new data, making sure, that the changes you make to encompass the new data does not result in deterioration of the analysis of previous data. This is also the lesson to be learned from the relative succes of constraint grammar, compared to rule-based methods, on unrestricted text.

The above is not to say that machine-learning is not a viable approach in other respects. This may be to improve the underlying constraint grammar analysis (using source language data), the transfer rules (using bilingual data) or the generation component (using target language data).

The final and possibly most radically note regarding the extraction of grammar rules: while the grammar-rules/lexical items which are extracted covers Danish, the rules that the system needs, will be English rules for generation. The system thus presupposes identity or a rule-like relation between the actual grammars of the source and target language, except for in the core, where idiosyncrasies may be handled; at least when they are Danish and English.

The structure of the MT algorithm The MT-system takes CG-analyses, with their dependencies explicated, as input in the form of a list of Prolog `word/9` formats described above. To prepare processing, each element of this list, i.e. each word of the sentence, is paired up with a variable. These variables are to be bound to a representation of 'what the word refers to' or the external argument of the semantics for the word. These structures are what I call the the semantic indeces of the words and I will describe these further below. Another preperation to processing is that the lexical entries, that were established for the previous sentence(s) are deleted from memory.

Then each word is 'bagified': each `word/9`-structure is translated to the corresponding leaf-node analysis, which would have resulted from a classical unification grammar. The purpose of this structure is to be fed into the 'double transfer'-system, which I describe on page 7, Because this system is making abstractions over the leaf-node representations, not all the information that you

³PRELIMINARY NOTE: THE FOLLOWING THREE PARAGRAPHS ADDED END OF SEPT. 2004

would find on the leaf-nodes in a classical unification grammar is included in my representations: I do not want to use time to extract information, that is thrown away in the next step of the process anyway. Apart from information which is strictly theory-dependent, like gapping, and information which is absent in the CG-analysis, like case of nouns, the information which is not extracted is for example the specific syntactic form of the dependents. The semantic form, including the lemma, *is* extracted, though, since this is information which is needed in the content strand of 'double transfer'.

This analysis is a sign with two parts: syntax and semantics. The syntax is a prolog-list, whose first element is the category of the word (np, vp etc.) and the two following are typically difference lists for gapping (which will be empty) and relativization (which will be empty, except for in the relative pronoun). Following that, is other syntactically relevant information, ending with a subcat-list (i.e. the valency), if this is relevant for a category of this type. These syntax representations are not 'proper Prolog lists' but rather pseudo-terms, as they will always have a fixed length and order, depending on the category of the word. The term-like structure is an advantage in simple systems as it is a simple structure. In a system like the present a feature-value based system (using what in other programming language is called a hash) or alternatively always accessing the term through auxiliary predicates, would possibly be superior, since you need not be aware of the order of the features and new features may be added when needed, without extending the length of the term-list everywhere it occurs in the program. Both these strategies will make running the program slightly slower, though.

The semantics is a representation of a predicate logic predication. This predication is represented (originally by Trujillo) in a somewhat particular way: as comma-separated list in round brackets with (what corresponds to) the predicate, typically the lexical item itself, as the very last element. The other arguments will be *semantic indices*, representations of the (referents of the) complements except for the first which will be 'the external argument', a representation of the referent of the word. A semantic index for an individual (the referent of a noun) will look like this: `ind(Var,ref,agr(3,sg,neut))`. The 'Var' is a Prolog variable which is never bound; it is used to distinguish the semantic index from semantic indices referring to other individuals/events. I call this variable the *referential index*. In Trujillo's simple original grammar 'ref' was not used in contradistinction to anything else. I use two other values for that spot: 'rel' for relative pronouns and 'app' for appositions. These two are characteristic by being co-referential, but still (in different ways) distinct from what they are co-referential to. The agr-feature for English does not contain gender, but only person and number. Number may be singular, plural or mass. The other type of semantic index (apart from individual) is the event. It contains a referential index, tense (in Danish combined with voice), agreement and, for Danish, topic, though this argument is only bound in the pilot project.

For words with potential dependents an important part of 'bagifying' is to establish the dependents and represent them properly in the subcat-list and in the semantics. This is done by a predicate that, depending on the type of word, goes through all types of dependents for this word, searches for them in

the sentence, and translates what is found to the proper format.

The process of extracting the subcat-frame is more 'sloppy' in this system than it would have to be in a proper grammar extraction system. Firstly: In a concrete sentence a concrete verb will take a concrete, fully specified, noun-phrase as subject. The valency of the verb is more abstract, but how abstract is impossible to know from just one instance, the information on the subcat-list will thus be typically just the category and some extra information that may be necessary as case, form of infinite verbs, that the subject-NP is a dummy 'there' etc. The valency extraction system does not work with lexical types and does not conjecture a more abstract underlying valency, from which the actual valency could be derived. A prime example where this gives lexical items that are theoretically 'improper' is in connection with passives, where the valency registered is the valency of the passive form and not its active counterpart. Finally, sometimes structures that would properly be handled as modification or processes on the subcat-list by a lexical rule, are handled directly in the lexical item, making the subcat-list longer. This is due to the fact that in a typical rule-based system the grammar rules need to be (overly?) abstract to cover as much ground as possible, while a high degree of explicitness is in its place in this usage of rules, since we want as precise a description as possible of the specific construction, not its place in a grander theory of language.

Transfer and generation The transfer and generation algorithm is identical to the one used in the pilot project, using the code from Trujillo (1999)⁴ as described in the description of the pilot project 'Three strategies for machine translation'. The only difference is that lines of the form 'Le <--> Ld', that looks up the Danish lexeme in the bilingual dictionary, are replaced with 'dbl_trf(DL, Le <--> Ld)', where DL is the Danish sign, ie. the syntax-semantics analysis for the word to be lexically translated. This may either be satisfied by calling the bilingual dictionary of the original system, ie. exactly as if the code had not been changed, or as 'double transfer' as described below. This method presupposes, that if the word is in the bilingual dictionary it is also in the monolingual English dictionary, a relatively innocent demand, and, more problematically, that the lexical sign in the English dictionary will be correct and appropriate, with the appropriate subcat-frame and even category, since the original bilingual dictionary does not distinguish between the different wordclasses.

This is where the interplay takes place between the two systems: the hand-coded transfer system and the 'ad hoc' system, which takes effect when the hand-coded system does not cover. The interplay is very hard to test though, because the hand-coded system, as it is now, is so sparse that it will never cover the lexicon and structure of a random sentence. In a sense, though, this just shows how efficient the described system is compared to classical hand-coded systems.

⁴PRELIMINARY NOTE: PROPER REF

Double transfer In the cases where the double transfer takes place, ie. where there are no translation of the word string in the original system, the first thing that happens is that string translation is done on the basis of the VISL bilingual dictionary. The original VISL bilingual dictionary has been automatically compiled (by a small Perl program) from the format: 'laborant s n r _#lab=technician' to the format 's:>laborant<*->'lab=technician', where the 's' in both notations is an abbreviation of 'substantiv', Danish for noun. The bilingual lexicon thus contains information on the word class, and for this reason this information has to be extracted from the sign and translated to this new notation. If the word does not occur in the VISL bilingual lexicon the system (at present) fails,⁵ except if it is a proper noun, which is taken to be the same in English if no explicit translations exist.

The process described in the last paragraph is in principle exactly the same as in the hand-coded system, but using a different lexicon.

Double transfer also contains a translation of the lexical sign with the underlying premis that if two strings are translations of each other then the lexical signs, that they are strings for, will have very similar or corresponding features in the two languages, including the same subcategorization. This is of course far from always true, but it is a useful hypothesis for the task at hand and where the languages in question are Danish and English.

Two processes are conflated in this translation process: on the one hand the categories of the two languages and the notation used in the particular grammars for the two languages are translated and on the other hand the lexical item is abstracted from being a concrete use of particular word⁶ to the conjectured restrictions on the use of the word, that you would find in a lexical item in a lexicalized grammar.

For the syntactic side of the sign eight simple schemata performs this transformation, which is partly recursive, since indices, elements on the subcat-list a.o. has to be transformed as well. One important change is the order in the subcat-list, since the English grammar has subjects at the end, as it is the last element to be combined with the verb, while in Danish, being a verb-second language, it is the first, and thus belong in the very beginning. For the semantic side there is a generalization consisting in removing extraneous information from the semantic indices, ie. everything but the referential index, and a translation that ensures the format of the indices are correct for English, removing topic-information for events and gender-information form agreement. The lexical item is then included in the database using `assert/1`, where it will be indistinguishable from a hand-coded lexical item.

The principal idea of the whole system is thus, that it behaves as much as a 'classical system' as possible, except when (as is normally the case) the lexical item that the sentence contains is not in the system, in which case it is conjectured. In this process the analysis is first being translated into a representation akin to the unification grammar notation, that the pilot project

⁵A small 'pre-analysis' module exists that may analyse the input for this problem before it is fed to the system, and thus contravene it.

⁶As described and discussed on page 5.

uses. After this translation, the information is split into two strands. One is the text-dependent, representing the conceptual structure of the input sentence, the information you need to get transferred from one language to another. This is the same as in the pilot-project. The other strand is the linguistic resources, the grammar of the Danish language in the form of lexical entries, that enables the content to be expressed in Danish. These linguistic resources are then 'translated' into English, presupposing that English grammar is to a large extent parallel to Danish grammar, and that non-parallelisms may be caught either in:

1. the grammar rules proper (i.e. the phrase structure rules in contradistinction to the lexical items)
2. the translation rules for the lexical items
3. hand-coded lexical items

In the grammar rules are treated among other phenomena the difference between morphological Danish and syntactic English determinateness and part of the verb-secondness of Danish. In the translation-of-rules process are covered among other things the presence/absence of morphological features, such as gender of Danish nouns and voice of Danish verbs, the topic-argument in Danish event-semantics, besides the morphological marking of definiteness and movement of subjects in the subcat-list. The third strategy, hand-coded lexical items, which is the only strategy available (together with transfer rules) in classical systems, is not yet used in my system. This is due to two facts: firstly, that each hand-coded rule has a relatively small coverage, and the quality of the rule has to be high to avoid that it blocks the back-off system.

4 Problems

This section is a list of some of the obstacles I have had in my work so far, that will also be obstacles in my further work. They are some problems of the system that I will very shortly list in random order, and without any pretence that they are all serious problems or that it is an exhaustively list. Many of the problems will be formulated as hindsights:

The pilot project is too small The interplay between the classic system and the ad hoc system has cannot be tested (and has to some extent been ignored) because the classic system never kicks in.

The ad hoc system is too small The ad hoc system which was meant to be a robust all-covering system only covers three elect sentences and extending it is a slow process. This is due to the fact that the translation procedure from CG-structures to signs is more heterogenous than expected. Thus the construction of the translation system needs the application of a lot of specific linguistic knowledge. Other reasons are the 'one phenomena at a time' approach I have taken, especially in designing (?) the notation.

Messy code Code from alternative definitions of subproblems is used without cleaning up, resulting in lots of lazy code. The small problems are left by the wayside. When a solution to a problem is found, it is not 'institutionalized', and when it is, I often forget what the method for solving this problem is.

No prior linguistic theory The grammar is extended at a problem-basis: can it handle the first sentence, no?, extend it; can it handle the second. Neither prior to the task, nor as problems are solved, are the broader picture seen. In the final thesis this may be remedied by 'looking ahead' at some of the problems that crops up in yet untreated data, explaining why they are problems, and giving ideas about how to solve them.

The sign-notation The 'term-type' sign notation is difficult to work with as it is difficult to extend (each step of the process has to be modified), and you have to remember the order and content of each argument. The semantics, with one simple predication for each word is too restrictive.

Modularity The modularity, which should have been an advantage, has turned out problematic, since each grammatical phenomenon and each notational solution has to be treated in all/most of the modules. This also is due to a bad pre-definition and constant change of what the interfaces are.

Time All/most of the above problems have to do with lack of time. I have from a very early stage felt in need of time, and have thus prioritized 'getting at coding' and 'don't mend it unless it is broken' over planning. This also means I have felt little need to look at any theory (combined with a problem with finding relevant theory).