

Grammar Engineering — Winter 2005 (Final Exercise)

High-Level Goals

- Work as independent grammar engineers, building a toy MT system.
- Explore the choice between interlingua- vs. transfer-based approaches.

1 Setting up for Machine Translation (0 Points)

- Transfer-based machine translation is a fast-moving field. The local LKB installation on the Meyer Windows machines (or your private laptops) is too out-of-date to give us the expressive power we will need to build an Esperanto–English machine translation system. Thus, for this exercise, we will have to work remotely on the Sweet Hall Linux machines (we will try to make an updated LKB binary for Windows available before the weekend).
- Launch the *bash* shell from the *Grammar Engineering* group; in *bash*, start a local X server by typing

```
startx
```

A new window with a(nother) shell should pop up. Use this window to connect to the Sweet Hall Linux cluster (using your own SUNet user id, naturally):

```
ssh -Y oepen@raptor.stanford.edu
```

- Obtain our starting package, essentially the model solutions for assignments # 4 and # 6, plus some additional configuration options and a skeleton transfer grammar:

```
cvs checkout mt
```

- Throughout this exercise, you will be working on two complete grammars, one in the new ‘*mt/esperanto/*’ sub-directory, another in ‘*mt/english/*’. Also, we will have two independent instances of *emacs* and the LKB, one processing Esperanto, the other processing English. As there will be parallel versions of all of the grammar files (e.g. a ‘*types.tdl*’ et al. in each of the two grammars), there is rich potential for confusing yourself here. Make sure to exercise good engineering discipline, for example assigning dedicated screen ‘divisions’ to the Esperanto and English universes; likewise, it will probably be useful to reserve one *emacs* for each language exclusively, so as to maintain a slight chance of maintaining your mental *saneco*.
- A few more notes of encouragement. Building a functional MT application is at least as much an engineering challenge as it requires good linguists. As you modify our two grammars—harmonizing lexicons and semantics (in an interlingua spirit) or adding transfer correspondences (using the MRS rewrite facilities of the LKB)—watch the Lkb Top and Lisp console windows (typically the *emacs* buffer called **common-lisp**) closely. Also, try to test changes individually (e.g. confirming that parsing results produce correct MRSs or testing generation mono-lingually first) before attempting end-to-end translation.

2 A Basic MT System (40 Points)

- Before we can expect to translate between Esperanto and English, we will need to harmonize the two grammars in terms of the choice of ‘domain’. Add at least ten of the nouns, verbs, and adjectives from our Esperanto grammar to the English lexicon; if you would rather work in the domain of violence among animals, consult a bilingual English–Esperanto dictionary to extend our Esperanto lexicon.
- We will typically invoke translation as a post-processing step on a parsing result, i.e. one of the little trees in the parsing results summary window. Assuming you have parallel vocabulary for, say, *la bela knabo dormas*, invoke the command *Rephrase* from the pop-up menu on the parse tree. Asking the LKB to ‘rephrase’ the semantics of a parsing result will invoke two steps: first, the source-language MRS is sent through the transfer component, aiming to map it into an MRS suitable for generation in the target

language; as we have chosen to use ‘interlingua’ predicate names for simple lexical entries, there should be no change to the MRS in this example. The resulting MRS(s) after transfer are displayed in a new window. Second, the transfer output is sent to the target language LKB, in this case the one loaded with the English grammar, for generation. Looking at the `*common-lisp*` buffer in our English universe, you should see messages somewhat like:

```
[12:05:04] translate(): read 1 MRS as generator input.  
[12:05:04] translate(): processing MRS # 0 (4 EPs).  
[12:05:04] translate(): 4 generation results.
```

- Once generation completes, another window will pop up to show the target language translations (which should be English for the current example). Make sure you can translate *the pretty boy* in both directions before moving on. Remind yourself of why we sometimes generate more outputs than we might have expected from the original input sentence.
- Throughout this exercise, it will occasionally be the case that the MRS output from transfer (once sent to the other LKB for generation) is ill-formed in some way, i.e. does not generate successfully. To debug generator failures, again, look at the `*common-lisp*` output buffer; a message like

```
[12:04:32] translate(): error ‘unknown predicates: |"adorn_rel"|’.
```

for example, would indicate that the generator input contained a predicate that is unknown to the target grammar; this could simply be a lack of lexical coverage or, as we move to more complex examples, indicate a specific way of using a predicate that is incompatible with its specification in the target language. This latter case, for example, can be triggered by a mismatch in value types on one of the `ARG0`, `ARG1`, et al. roles or even a binding across multiple roles that is invalid according to the target grammar. To debug generator failures (should you experience any; we know some grammar engineers that do frequently), expand the LKB Top window (through the `Options|Expand` menu command) to get access to the `Generate|Redisplay MRS` menu entry. Parsing the expected generator output using the target grammar and comparing its MRS to what was sent to the generator as input should enable you to identify semantic mismatches and make corrections (in either grammar) accordingly.

- It is quite possible (likely, some would say) that you will discover deficiencies in our ‘model’ solutions as you work on machine translation. More engineering makes better grammars. As you discover overgeneration in either grammar, use your grammar engineering experience to make the necessary corrections.
- Next, we will enrich our semantics to include information about, minimally, tense and number. Consider adding a property `TENSE` to events and provide a small hierarchy of available tense values (including *present* and *past*, obviously). We will assume that index properties—i.e. features defined on sub-types of *index*—will be valid cross-linguistically. Thus, as you add features into the semantic variables, make sure to maintain parallelism across the two grammars; sending an MRS with invalid or undefined index properties to the generator will almost certainly fail to generate (and likely produce wild error messages). Once you have added `TENSE` and its appropriate value types in `types.tdl` make sure that the inflectional rules actually specify correct tense values (note that the two grammars follow a different style in how they specify lexical rules). For each language, parse a few sentences and inspect the MRSs for correctness before you ask the system for translations. Do the equivalent for number, i.e. make sure that number distinctions are reflected in the semantics. Maybe just introduce an index property `NUMBER` at this point, if you are eager to improve translation quickly. We will encourage you to consider some further harmonization of the encoding of person and number information across the two grammars later in this assignment, ultimately aiming to move agreement constraints into the semantics completely.
- Now, finally, we will introduce transfer rules for mismatches between the two languages that would be hard to accommodate in an interlingua-based fashion: look at the type definition of *adjective_negation_mtr* and the one transfer rule instantiating this correspondence type in the new file `eoen.mtr`. Find an Esperanto example that you expect to make use of the *ugly* MRS transfer rule (MTR), parse it, and invoke the `Rephrase` command on it. Make sure to understand the MRS that you see post-transfer and then ensure successful translation into English, possibly adding another lexical entry. At this point, we will not aim for reversibility of transfer, i.e. will be content to translate from Esperanto to English only. Add a few more transfer rules for decomposed Esperanto adjectives and confirm that they translate successfully.

- Using the *adjective_negation_mtr* types as examples, create additional transfer correspondence types *feminine_noun_mtr*, *adjective_nominalization_mtr*, and *verb_nominalization_mtr*. Create transfer rules using these new types in `'eoen.mtr'` and populate each pattern with at least two examples. As always, you will have the alternate choice of semantic decomposition in the English lexicon, i.e. allowing some lexemes to have more than one EP in their RELS list lexically. Reflect on these alternatives, maybe experiment with the interlingua approach for at least one of the above correspondence types, and summarize your findings in one or two paragraphs. What are the (practical) advantages and disadvantages of either encoding?

3 Optional: More Machine Translation (60 Points)

- Continue improving bidirectional coverage in our toy MT system; the field is wide open for you. Using the techniques exemplified so far, pick additional phenomena for translation and then either harmonize the two grammars further or add transfer rules, preferably enabling translation in both directions. For each such phenomenon, we will award between five and twenty points, depending on (a) the complexity of the problem, (b) the correctness and adequacy of your solution (including whether or not translation works bi-directionally), and (c) the amount of documentation provided to us. Note that, without documentation, we will have no way of knowing which phenomena you have targeted in addition to the ones from the first part of this exercise. The following paragraphs offer some vague ideas about candidate phenomena, but there are many more, of course.
- The MT research community has made considerable progress in the past few days, so make sure to bring your grammars in-line with some (relatively minor) changes we have made since you obtained the original starting package. The command

```
cvns update mt
```

will merge updates we have applied to the master source repository into your local copy, but CVS will never overwrite any local changes you have made. Study the output of the `update` command and maybe consult the CVS on-line documentation (`'man cvs'` from the shell prompt), in case you want to know more about the procedure. In a nutshell, for each file, 'M' means you have local modifications, 'P' and 'U' indicate that you receive an update from the central repository, and 'C' suggests trouble (a conflict between changes we made and your own modifications).

- *More Index Properties* Create a type hierarchy for *pernum* values that can be used cross-linguistically, i.e. in both our grammars. Maybe start from the (English) hierarchy suggested on our slide # 110 and then see whether you can express the AGR distinctions made in Esperanto within this hierarchy, or whether you want to insert additional nodes (e.g. *sg*, *pl*, and maybe others). Use the `View | Type hierarchy` command to inspect the resulting hierarchy and make sure there are no 'anonymous' greatest lower bound types. If necessary, give interpretable names to those types, should the LKB compiler need to introduce any. Replace `NUMBER` on *object* with a new feature `PN` and supply appropriate values for (at least) all pronouns in the lexicon (if need be, consult an on-line grammar of Esperanto to refresh your understanding of its pronoun system). Ensure that generation within the Esperanto universe does not result in spurious outputs when using the MRS of a sentence involving pronouns as the generator input; if necessary, add additional 'semantic' properties to sub-types of *index*. Now add personal pronouns to the English grammar and test translation.
- *Spring Cleaning* As experienced grammar engineers, you must be irritated at the considerable amount of redundancy that holds between the two grammars: many of the type definitions—i.e. large parts of the linguistic hierarchy—are more or less identical in both versions of our `'types.tdl'` file. Our (updated) starting package already provides you with an (empty) file `'common.tdl'` in the `'mt/'` directory, i.e. the parent directory to the two grammars; `'common.tdl'` is loaded by both grammars (see the `'script'` files in each grammar-specific directory to confirm this), thus it provides a way of sharing type definitions across the two grammars. Review the file `'types.tdl'` from both grammars and move some 70 or so definitions into `'common.tdl'`. In doing so, you will create a much stronger parallelism between the two grammars and may have to 'weed out' some unwarranted differences, for example the variation in spelling for **bool** vs. *boolean* and `--PM` vs. `MODIFIED`. This type of low-level harmonization, unfortunately, is something that grammar writers end up doing more frequently than they should have to, especially in a machine translation set-up. A slightly more substantial (but equally unnecessary) diversion between

the two grammars in our starting package is their treatment of ORTH, viz. as just a **string** in Esperanto (where it only occurs within the lexicon) and as a difference list in English, where ORTH is accumulated from the daughters on phrases. Veteran grammar writers and students of grammar engineering alike hate difference list concatenation with great passion. Assume that ORTH is only needed on sub-types of *lex-item* and rework the English lexicon and type definitions appropriately. Also, fully eliminate the appending of ORTH on phrase types and celebrate appropriately. In doing so, you will have made a change to the overall feature geometry (for English) that needs to be declared to the LKB. Edit ‘globals.lsp’ in the English grammar and change the definition of **orth-path** to read as follows:

```
(defparameter *orth-path* '(ORTH))
```

- *Semantic Agreement* Eliminate the (syntactic) feature AGR, making use of our enriched semantic variables (aka indices). By and large, the binding of variables in the semantics should already have the intended effect, but in a few cases you may have to enrich a type definition or lexical rule. Obviously, you have used the batch parsing machinery (and occasional interactive generation and translation) frequently to test each step in changing your grammars. To make comparison to earlier versions a little easier (a technique called *regression testing*, consider keeping a copy of an earlier ‘all.results’ file and using the shell *diff* command to more easily compare newer results to how things used to be.
- *Possessive Determiners* Add possessive determiners like *my*, *her*, et al. to the English grammar. There is not much point in trying to derive these from personal pronouns (as we do in Esperanto), so consider just adding them to the lexicon with an appropriate part of speech. However, the derived semantics we get in Esperanto seems highly adequate, so arrange for the new lexical entries to have a decomposed semantics, i.e. involving a combination of *possessive_rel* plus *pron_rel*, in English too.
- *Relational Nouns* Family relations are commonly analysed as relational nouns, where, say, *brother* takes an optional PP[of] complement. Assuming that the preposition itself is semantically vacuous, then *[the] brother of [his] father* would have a semantics much like:

```
brother_rel [ARG0  $x_1$ , ARG1  $x_2$ ], father_rel [ARG0  $x_2$ , ARG1  $u$ ]
```

Review our earlier discussion of optional complements and make sure you can analyze both *the brother of my father sleeps* and just *the brother sleeps*. As always, avoid spurious ambiguities, i.e. alternate parse trees that do not correspond to differences in interpretation.

- *Transfer Ambiguity* At the 2003 International Esperantora, the annual conference of eccentric linguists, there was a special session on the interpretation of *knab*. Although it is commonly translated as English *boy*, several scholars argued that a far more appropriate translation would be as *young man*. While the scientific discourse proceeds, we should move away from our interlingua treatment of *knab* and assign it its own language-internal predicate name, say *knab_rel*. Make this change in the lexicon and then add a transfer correspondence type that maps a noun semantics into another noun semantics; make this MTR type optional by letting it inherit from *monotonic_omtr*. Add a transfer rule to ‘eoen.tdl’ to translate *knab* as *boy*, so as to accommodate the traditionalist view. Next, add an MTR type to map the semantics corresponding to a single noun into the MRS of the combination of an adjective plus a noun. Again, instantiate this transfer correspondence type with an MTR instance, in order to translate *knab* as *young man*. Observe how the *Rephrase* command can result in multiple transfer outputs (the *Previous* and *Next* buttons on the transfer MRS viewer allow navigation through the set of transfer outputs) and how each, in turn, is then sent to the target language generator. Do something in the same spirit for the feminine use of *knab* and order your transfer rules correctly, i.e. according to optionality vs. non-optionality and specificity of input descriptions. Next, try to optionally translate *onkl* as *brother of my mother* or *brother of my father*. Imagine the words *uncle* and *aunt* were not available in English; what would be the resulting challenge for an Esperanto–English machine translation system?

Submit your results in email to Dan and Stephan by noon on Friday, March 11.