Published on *Multilingual Online Translation* (http://www.molto-project.eu)

D10.2 MOLTO web service, first version

Contract No.:	FP7-ICT-247914
Project full title:	MOLTO - Multilingual Online Translation
Deliverable:	D10.2 MOLTO web service, first version
Security (distribution level):	Public
Contractual date of delivery:	M3
Actual date of delivery:	2 June 2010
Type:	Prototype
Status & version:	Final
Author(s):	Krasimir Angelov, Olga Caprotti, Ramona Enache, Thomas Hallgren, Inari Listenmaa, Aarne Ranta, Jordi Saludes, Adam Slaski
Task responsible:	UGOT [1]
Other contributors:	UPC [2], UHEL [3]

ABSTRACT

This phrasebook is a program for translating touristic phrases between 14 European languages included in the MOLTO project (Multilingual On-Line Translation): Bulgarian, Catalan, Danish, Dutch, English, Finnish, French, German, Italian, Norwegian, Polish, Romanian, Spanish, Swedish. A Russian version is not yet finished but will be added later. Also other languages may be added.

The phrasebook is implemented by using the GF [4] programming language (Grammatical Framework). It is the first demo for the MOLTO project, released in the third month (by June 2010). The first version is a very small system, but it will extended in the course of the project.

The phrasebook is available as open-source software, licensed under GNU LGPL, at http://code.haskell.org/gf/examples/phrasebook/ [5].

1. Purpose

The MOLTO phrasebook is a program for translating touristic phrases between 14 European languages included in the MOLTO project (Multilingual On-Line Translation):

- Bulgarian, Catalan, Danish, Dutch, English, Finnish, French, German, Italian, Norwegian, Polish, Romanian, Spanish, Swedish. A Russian version is not yet finished but is projected later. Other languages may be added at a later stage.

The phrasebook is implemented in the GF [4] programming language (Grammatical Framework). It is the first demo for the MOLTO project, released in the third month (by June 2010). The first version is a very small system, but it will be extended in the course of the project.

The phrasebook has the following requirement specification: - high quality: reliable translations to express yourself in any of the languages - translation between all pairs of languages - runnable in web browsers - runnable on mobile phones (via web browser; Android stand-alone forthcoming) - easily extensible by new words (forthcoming: semi-automatic extensions by users)

The phrasebook is available as open-source software, licensed under GNU LGPL. The source code resides in ftp://code.haskell.org/gf/examples/phrasebook/ [6]

2. Points Illustrated

We consider both the end-user perspective and the content producer perspective.

From the user perspective

- Interlingua-based translation: we translate meanings, rather than words
- Incremental parsing: the user is at every point guided by the list of possible next words
- Mixed input modalities: selection of words ("fridge magnets") combined with text input
- Quasi-incremental translation: many basic types are also used as phrases, one can translate both words and complete sentences, and get intermediate results
- Disambiguation, esp. of politeness distinctions: if a phrase has many translations, each of them is shown and given an explanation (currently just in English, later in any source language)
- Fall-back to statistical translation: currently just a link to Google translate (forthcoming: tailor-made statistical models)
- Feed-back from users: users are welcome to send comments, bug reports, and better translation suggestions

From the programmer's perspective

- The use of resource grammars and functors: the translator was implemented on top of an earlier linguistic knowledge base, the GF [4] Resource Grammar Library
- Example-based grammar writing and grammar induction from statistical models

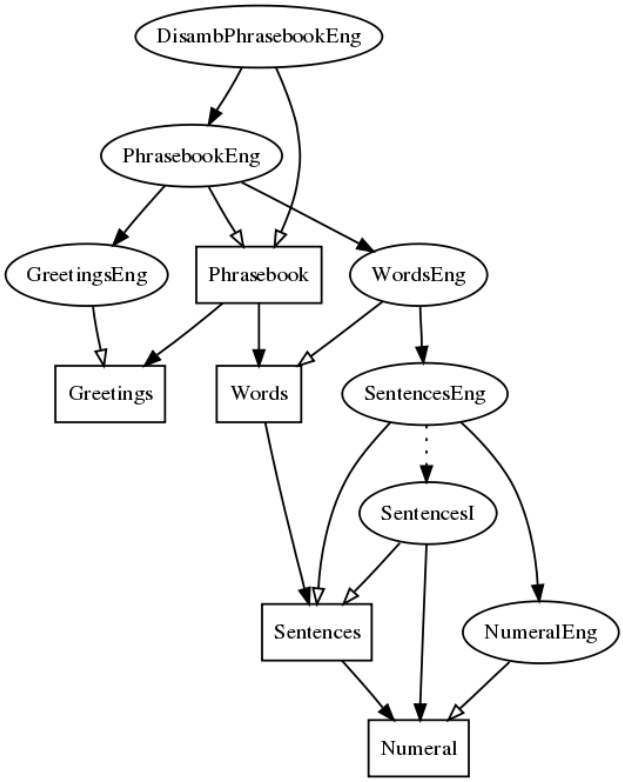
(Google translate): many of the grammars were created semi-automatically by generalization from examples
- Compile-time transfer especially, in Action in Words: the structural differences between languages are treated at compile time, for maximal run-time efficiency
- The level of skills involved in grammar development: testing different configurations (see table below)
- Grammar testing: use of treebanks with guided random generation for initial evaluation and regression testing

3. Files

The phrasebook is available as open-source software, licensed under GNU LGPL. The source code resides in http://code.haskell.org/gf/examples/phrasebook/ [5]. Below a short description of the source files.

Grammars

- `Sentences`: general syntactic structures implementable in a uniform way. Concrete syntax via the functor `SencencesI`.
- `Words`: words and predicates, typically language-dependent. Separate concrete syntaxes.
- `Greetings`: idiomatic phrases, string-based. Separate concrete syntaxes.
- `Phrasebook`: the top module putting everything together. Separate concrete syntaxes.
- `DisambPhrasebook`: disambiguation grammars generating feedback phrases if the input language is ambiguous.
- `Numeral`: resource grammar module directly inherited from the library.



The module structure image is produced in GF [4] by

```
> i -retain DisambPhrasebookEng.gf
> dg -only=Phrasebook*,Sentences*,Words*,Greetings*,Numeral,Nu
> ! dot -Tpng _gfdepgraph.dot > pgraph.png
```

Ontology

The abstract syntax defines the **ontology** behind the phrasebook. Some explanations can be found in the ontology document [7], which is produced from the abstract syntax files `Sentences.gf` [8] and `Words.gf` [9] by `make doc`.

Run-time system and user interface

The phrasebook uses the PGF server [10] written in Haskell and the minibar library [11] written in JavaScript. Since the sources of these systems are available, anyone can build the phrasebook locally on her own computer.

4. Effort and Cost

Based on this case study, we roughly estimated the effort used in constructing the necessary sources for each new language and compiled the following summarizing chart.

Language	Language skills	GF [4] skills	Informed development	Informed testing	Impact of external tools	RGL Changes	Overall effort
Bulgarian	###	###	-	-	?	#	##
Catalan	###	###	-	-	?	#	#
Danish	-	###	+	+	##	#	##
Dutch	-	###	+	+	##	#	##
English	##	###	-	+	-	-	#
Finnish	###	###	-	-	?	#	##
French	##	###	-	+	?	#	#
German	#	###	+	+	##	##	###
Italian	###	#	-	-	?	##	##
Norwegian	#	###	+	-	##	#	##
Polish	###	###	+	+	#	#	##
Romanian	###	###	-	-	#	###	###
Spanish	##	#	-	-	?	-	##
Swedish	##	###	-	+	?	-	##

Legend

Language skills

- - : no skills
- # : passive knowledge
- ## : fluent non-native
- ### : native speaker

GF [4] skills

- - : no skills
- # : basic skills (2-day GF [4] tutorial)
- ## : medium skills (previous experience of similar task)
- ### : advanced skills (resource grammar writer/substantial contributor)

Informed Development/Informed testing

- - : no
- + : yes

Impact of external tools

- ?: not investigated
- - : no effect on the Phrasebook
- # : small impact (literal translation, simple idioms)
- ## : medium effect (translation of more forms of words, contextual preposition)
- ### : great effect (no extra work needed, translations are correct)

RGL changes (resource grammars library)

- - : no changes
- # : 1-3 minor changes
- ## : 4-10 minor changes, 1-3 medium changes
- ### : >10 changes of any kind

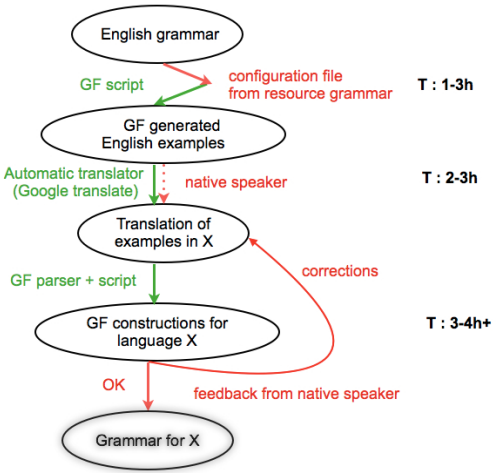
Overall effort (including extra work on resource grammars)

- # : less than 8 person hours
- ## : 8-24 person hours
- ### : >24 person hours

5. Example-based grammar writing process

The figure presents the process of creating a Phrasebook using an example-based approach for a language X, in our case either Danish, Dutch, German, Norwegian, for which we had to employ informed development and testing by a native speaker, different from the grammarian.

Remarks : The arrows represent the main steps of the process, whereas the circles represent the initial and final results after each step of the process. Red arrows represent manual work and green arrows represent automated actions. Dotted arrows represent optional steps. For every step, the estimated time is given. This is variable and greatly influenced by the features of the target language and the semantic complexity of the phrases and would only hold for the Phrasebook grammar.



Initial resources :

- English Phrasebook
- resource grammar for X
- script for generating the inflection forms of words and the corresponding linearizations of the lexical entries from the Phrasebook in the language X. For example, in the case of the nationalities, since we are interested in the names of countries, languages and citizenship of people and places, we would generate constructions like "I am English. I come from England. I speak English. I go to an English restaurant" and from the results of the translation we will infer the right form of each feature. In English, in most cases there is an ambiguity between the name of the language and the citizenship of people and places, but in other languages all three

could have completely different forms. This is why it is important to make the context clear in the examples, so that the translation will be more likely to succeed. The correct design of the test of examples, is language dependent and assumes analysis of the resource grammar, also. For example, in some languages we need only the singular and the plural form of a noun in order to build its GF [4] representation, whereas in other languages such as German, in the worst case we would need 6 forms which need to be rendered properly from the examples.

- script for generating random test cases that cover all the constructions from the grammar. It is based on the current state of the abstract syntax and it generates for each abstract function some random parameters and shows the linearization of the construction in both English and language X, along with the abstract syntax tree that was generated.

Step 1 : Analysis of the target grammar

The first step assumes an analysis of the resource grammar and extracts the information needed by the functions that build new lexical entries. A model is built so that the proper forms of the word can be rendered, and additional information, such as gender, can be inferred. The script applies these rules to each entry that we want to translate into the target language, and one obtains a set of constructions.

Step 2 : Generation of examples in the target language

The generated constructions are given to an external translator tool (Google translate) or to a native speaker for translation. One needs the configuration file even if the translator is human, because formal knowledge of grammar is not assumed.

Step 3 : Parsing and decoding the examples with GF [4]

The translations into the target language are further more processed in order to build the linearizations of the categories first, decoding the information received. Furthermore, having the words in the lexicon, one can parse the translations of functions with the GF [4] parser and generalize from that.

Step 4 : Evaluation and correction of the resulting grammar

The resulting grammar is tested with the aid of the testing script that generates constructions covering all the functions and categories from the grammar, along with some other constructions that proved to be problematic in some language. A native speaker evaluates the results and if corrections are needed, the algorithm runs again with the new examples. Depending on the language skills of the grammar writer, the changes can be made directly into the GF [4] files, and the correct examples given by the native informant are just kept for validating the results. The algorithm is repeated as long as corrections are needed.

The time needed for preparing the configuration files for a grammar will not be needed in the future, since the files are reusable for other applications. The time for the second step can be saved if automatic tools, like Google translate are used. This is only possible in languages with a simpler morphology and syntax, and with large corpora available. Good results were obtained for German and Dutch with Google translate, but for languages like Romanian or Polish, which are both complex and lack enough resources, the results are discouraging.

If the statistical oracle works well, the only step where the presence of a human translator is needed is the evaluation and feedback step. An average of 4 hours per round and 2 rounds were

needed in average for the languages for which we performed the experiment. It is possible that more effort is needed for more complex languages.

Further work will be done in building a more comprehensive tool for testing and evaluating the grammars, and also the impact of external tools for machine translation from English to various target languages will be analysed, so that the process could be automated to a higher degree for the future work on grammars.

6. Future and ongoing work

Disambiguation
: Disambiguation grammars for languages other than English are in most cases still incomplete.

Lexicon extension
: The extension of the abstract lexicon in `Words` by hand or (semi)automatically for items related to the categories of food, places, and actions will result in immediate increase of the expressiveness of the phrasebook.

Customizable phone distribution
: Allow the ad-hoc selection of the 2^15 language subsets when downloading the phrasebook to a phone.

7. How to contribute

The basic things "everyone" can do are:

- complete missing words [12] in concrete syntaxes
- add new abstract words in `Words` and greetings in `Greetings`

The missing concrete syntax entries are added to the `WordsL.gf` files for each language **L**. The morphological paradigms [13] of the GF [4] resource library should be used. Actions (prefixed with `A`, as `AWant`) are a little more demanding, since they also require syntax constructors. Greetings (prefixed with `G`) are pure strings.

Some explanations can be found in the implementation document [14], which is produced from the concrete syntax files `SentencesI.gf` [15] and `WordsEng.gf` [16] by `make doc`.

Here are the steps to follow for contributors:

1. Make sure you have the latest sources from GF Darcs [17], using `darcs pull`.
2. Also make sure that you have compiled the library by `make present` in `gf/lib/src/`.
3. Work in the directory `gf/examples/phrasebook/` [5].
4. After you've finished your contribution, recompile the phrasebook by `make pgf`.
5. Save your changes in `darcs record .` (in the `phrasebook` subdirectory).
6. Make a patch file with `darcs send -o my_phrasebook_patch`, which you can send to GF [4] maintainers.
7. (Recommended:) Test the phrasebook on your local server: a. Go to `gf/src/server/` and follow the instructions in the project Wiki [10]. b. Make sure that `Phrasebook.pgf` is available to you GF [4] server (see project wiki). c. Launch `lighttpd` (see project wiki). d. How you can open `gf/examples/phrasebook/www/phrasebook.html` and use your phrasebook!

Finally, a few good practice recommendations:

- Don't delete anything! But you are free to correct incorrect forms.
- Don't change the module structure!
- Don't compromise quality to gain coverage: **non multa sed multum!**

8. Conclusions (tentative)

The grammarian need not be a native speaker of the language. For many languages, the grammarian need not even know the language, *native informants* are enough. However, evaluation by native speakers is necessary.

Correct and idiomatic translations are possible.

A typical development time was 2-3 person working days per language.

Google translate helps in bootstrapping grammars, but must be checked. In particular, we found it unreliable for morphologically rich languages.

Resource grammars should give some more support e.g. higher-level access to constructions like negative expressions and large-scale morphological lexica.

Acknowledgments

The Phrasebook has been built in the MOLTO project funded by the European Commission. The authors are grateful to their native speaker informants helping to bootstrap and evaluate the grammars: Richard Bubel, Grégoire Détrez, Rise Eilert, Karin Keijzer, Michał Pałka, Willard Rafnsson, Nick Smallbone.

MOLTO Phrasebook (version 1)

From: Eng To: All Del Clear Random

Help

a airplane airport am amusement an apple apples are
bad bank bar beer Belgian Belgium bike boring bread
Bulgaria Bulgarian bus by bye canteen car Catalan
Catalonia Catalonian center cheap cheers cheese chicken
church cinema coffee cold congratulations crown damn
Danish delicious Denmark disco do does dollar Dutch
eight eighteen eighty eleven England English euro excuse
expensive ferry fi fteen fi fty Finland Finnish fi sh fi ve
Flemish forty four fourteen France French fresh Friday
German Germany good goodbye happy hello help hospital
hotel how I is Italian Italy leu lev look meat milk
Monday museum my nice nine nineteen ninety NN no
Norway Norwegian one park pharmacy pizza pizzas please
Poland Polish post pound pub restaurant Romania
Romanian rouble Russia Russian salt Saturday school see
seven seventeen seventy shop six sixteen sixty sorry
Spain Spanish station subway Sunday supermarket suspect
Sweden Swedish taxi tea ten thank that the theatre
these thirteen thirty this those three Thursday toilet too
train tram Tuesday twelve twenty two university very
warm water Wednesday what where which wine yes you
your zloty zoo

Try Google Translate Feedback

Powered by GF [4], see doc [18]. We also have a mobile-enhanced version [19].

MOLTO Phrasebook Help

The user interface is kept slim so as to also be usable from portable devices, e.g. mobile phones. These are the buttons and their functionality:

- **To start**: klick at a word or start typing.
- **From**: source language
- **To**: target language (either a single one or "All" simultaneously)
- **Del**: delete last word
- **Clear**: start over
- **Random**: generate a random phrase
- **Google translate**: the current input and language choice; opens in a new window or tab.

The symbol `&+` means binding of two words. It will disappear in the complete translation.

The translator is slightly **overgenerating**, which means you can build some semantically strange phrases. Before reporting them as bugs, ask yourself: could this be correct in some situation? is the translation valid in that situation?

Source URL: http://www.molto-project.eu/node/1022

Links:
[1] http://www.molto-project.eu/University of Gothenburg
[2] http://www.molto-project.eu/Universitat Politècnica de Catalunya
[3] http://www.molto-project.eu/University of Helsinki
[4] http://www.grammaticalframework.org
[5] http://code.haskell.org/gf/examples/phrasebook/
[6] ftp://code.haskell.org/gf/examples/phrasebook/
[7] http://code.haskell.org/gf/examples/phrasebook/Ontology.html
[8] http://code.haskell.org/gf/examples/phrasebook/Sentences.gf
[9] http://code.haskell.org/gf/examples/phrasebook/Words.gf
[10] http://code.google.com/p/grammatical-framework/wiki/LaunchWebDemos
[11] http://www.cse.chalmers.se/~hallgren/minibar/about.html
[12] http://code.haskell.org/gf/examples/phrasebook/missing.txt
[13] http://code.haskell.org/gf/lib/doc/synopsis.html#toc78
[14] http://code.haskell.org/gf/examples/phrasebook/Implementation.html
[15] http://code.haskell.org/gf/examples/phrasebook/SentencesI.gf
[16] http://code.haskell.org/gf/examples/phrasebook/WordsEng.gf
[17] http://www.grammaticalframework.org/doc/gf-developers.html
[18] http://www.grammaticalframework.org/examples/phrasebook/doc-phrasebook.html
[19] http://www.grammaticalframework.org/demos/phrasebook