Dogs, frogs and lambs

Using Constraint Grammar for Celtic languages in Apertium

Francis M Tyers, Kevin Donnelly *

Abstract

We describe the integration of the **vislcg3** Constraint Grammar (CG) package into Apertium, a rule-based machine translation platform. We first give an outline of the steps necessary to achieve this integration, and then go on to consider some aspects of applying CG to the Celtic languages.

1 Introduction

Apertium is a machine translation platform originally developed for the Romance languages of the Iberian peninsula, but now being adapted for a variety of other languages, including Welsh [1] and Breton[1]. Working systems for 17 language pairs have now been released, and all the material (both software and data) is available under the Free Software Foundation's GPL license.

The **vislcg3** constraint grammar package was integrated into Apertium in order to meet a gap in the linguistic resources available - there was no tagged corpus of Welsh available under a free license with which to train the Apertium tagger, and yet the disambiguation of the Welsh text needed to be improved before it was fed into the statistical tagger.

After a description of the integration work, we discuss the use of the CG functionality to help Apertium distinguish between homonyms in the Celtic languages. We look at three types of homonyms: (1) those resulting from mutation, (2) interdependent homonyms, and (3) different meanings for the same part of speech. We then discuss the use of CG to "cast" POS tags for homonyms (e.g. to change a subject tag for a pronoun listed in the dictionary to an object tag for a homonymous pronoun not listed in the dictionary), and go on to look at how rules from another free software package for a Celtic language (Kevin P. Scannell's *An Gramadóir* grammar-checker for Irish) have been ported to CG. We end with some reflections on the importance of open licensing in the development of tools for marginalised languages.

2 Integration of vislcg3 into Apertium

2.1 The Apertium pipeline

(Todo: Summary of the Apertium pipeline and how vislcg3 fits into it)

*We are grateful to Gwenvael Jequel and ?anyone else? for help with aspects of this paper.

[1] `http://xixona.dlsi.ua.es/~fran/breton/traductor/index.php`

2.2 Tracing the application of CG rules

Detailed CG development depends on being able to trace which rules are being tripped during the disambiguation process. **vislcg3** already offers a mechanism for this via its `-trace` switch; the output of this is also available through Apertium.

(**Todo: Give command-line example**)

Agoriad[2], a web interface for **apertium-cy** (the Welsh version of Apertium), shows the output of each stage of the Apertium pipeline in a colour-coded tabular format. Figure 1 shows CG disambiguation for:

(1) a. **a oes lle yma?**

b. *is there space here?*

To keep track of the rules, they are each named using **vislcg3**'s naming option, with some attempt being made to keep the name mnemonic. For instance, the **R_oes_age** rule:

```
REMOVE:R_oes_age NC IF (-1 Vpart) (0 NC) (0 V);
```

will delete the noun **oes** (*age*) from a {verb, noun} cohort if preceded by a verbal particle, to leave the desired **oes** (*is*).

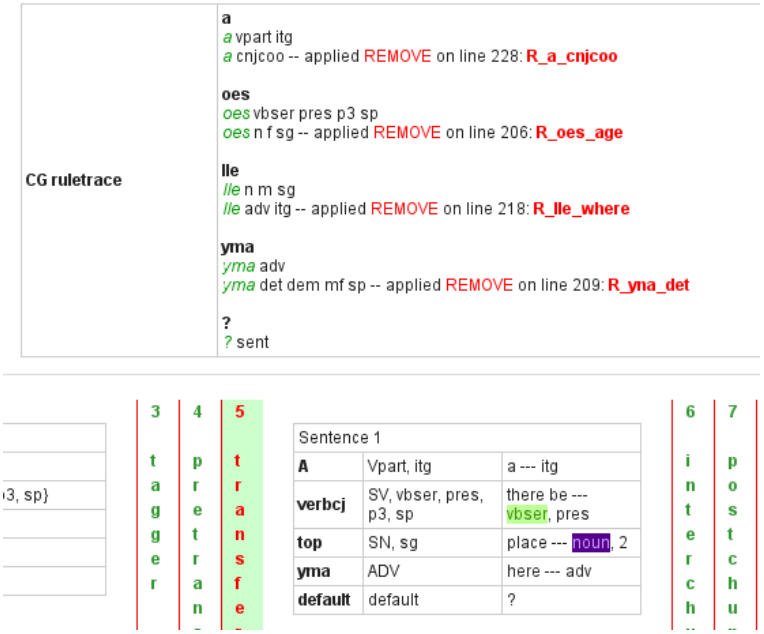

Fig. 1: The ruletrace function of Agoriad

3 Handling of mutation in Apertium

One notable characteristic of the Celtic languages is "mutation" - morphophonemic alteration of initial consonants. Consider the Welsh examples:

(2) a. **mae o'n marw**

[2] give web address and address of git repo

b. *he is dying*

(3) a. **mae o'n farw**

b. *he is dead*

Here, the change $m \rightarrow f$ signifies that the following word is an adjective and not a verb. In the text to be translated, these mutations have to be removed in order to get to the underlying lemma.

In Apertium's Welsh dictionary, the entry for each mutatable word contains a reference to a paradigm which deals with the initial variants. Thus, for **tad** (*father*) we have:

```
<e lm="tad"><par n="initial-t"/><i>ad</i>
<par n="aberth__n"/></e>
```

The paradigm in turn lists the various changes that can take place:

```
<pardef n="initial-t">
<e r="LR"><p><l>t</l><r>t</r></p></e>
<e r="LR"><p><l>d</l><r>t</r></p></e>
<e r="LR"><p><l>nh</l><r>t</r></p></e>
<e r="LR"><p><l>th</l><r>t</r></p></e>
<e r="RL"><p><l>t</l><r>t</r></p></e>
</pardef>
```

Each line relates to a specific mutation of the base form **tad**:
soft mutation: ***ei tad** → **ei dad** - *his father*
nasal mutation: ***fy tad** → **fy nhad** - *my father*
aspirate mutation: ***ei tad** → **ei thad** - *her father*

4 Disambiguating mutational homonyms

This process (which is purely grapheme-based, and takes no account of the morphological significance of the mutations) allows the de-mutated lemmas to be selected, but still leaves us with a set of ambiguous choices. CG allows the correct surface homonym to be chosen.

For instance, consider the collocation:

(4) a. **Y Bwrdd Glo**

b. *The Coal Board*

c. **The lock table*

Todo: The translation needs to be changed now that glo (coal) has been added.
Here, the second word, **glo**, gets returned by the dictionary as **glo** (*coal*), which is correct, and also as the soft mutation (**glo**) of **clo** (*lock*), which is not - a qualifier would never show a soft mutation after a masculine noun like **bwrdd** (*board*).

To deal with this, we use a CG rule as follows:

```
LIST SoftMut = ("<g.*>"ri "c.*"ri) ("<d.*>"ri "t.*"ri)
("<f.*>"ri "b.*"ri) ("<w.*>"ri "gw.*"ri)("<b.*>"ri "p.*"ri)
("<dd.*>"ri "d.*"ri) ("<f.*>"ri "m.*"ri);
LIST Adj = (adj);
SELECT SoftMut (-1 NF) (0 Adj);
```

This selects a soft-mutated adjective if it follows a noun which is feminine, but in the case above **bwrdd** is masculine, so the correct lemma, **glo** (*coal*), will be selected.

Another example is a much more common one. The surface form **chi** can be interpreted as either the second person plural pronoun (*you*), by far the commonest occurrence, or as an aspirate mutation of **ci** (*dog*). In earlier versions of **apertium-cy**, we had:

(5) a. **Gobeithio bod popeth yn iawn gyda chi.**
b. *I hope everything is well with you.*
c. **Hope be everything in very with a dog.*

The addition of the following rule:

```
REMOVE MAspC IF (NOT -1 ("ei")) (NOT -1 ("a" cnjcoo))
(NOT -1 ("chwech"));
```

means that all readings with a putative aspirate mutation will be dropped if they do not follow a word listed as causing an aspirate mutation. In more recent versions of **apertium-cy**, therefore, **chi** is properly translated:

**Hope be everything in very with you.*

Todo: Add rule for yn iawn to give a better translation.

In Breton, the verb **ober** (*to do*) can be used as an auxiliary to emphasise the action of the main verb, e.g.

(6) a. **c'hoari a ran**
b. *I am/will be playing* (literally "*it is playing that I will do*")

But another **ran** also exists, meaning *frog*, so that early versions of **apertium-br** (the Breton version of Apertium) translated this as **play frog*. This has been resolved with a rule to select the verb reading when it follows the relative verbal particle **a** (*which*):

```
LIST VpartRef = (vpart ref);
LIST Vbloc = (vbloc);
LIST Vblex = (vblex);
LIST Vbser = (vbser);
LIST TempsFin = (pri) (pii) (past) (fti);
SET Verb = Vbloc | Vblex | Vbser;
SET VerbFin = Verb | TempsFin;
SELECT VerbFin IF (-1 VpartRef);
```

Todo: Perhaps shorten this rule a bit by having less stacking. And is the | symbol correct - should it not be +?

Todo: Add something about the mutation trigger rules in the Breton CG.

5 Disambiguating interdependent homonyms

"Interdependent homonyms" can be defined as a collocation in which each element is ambiguous. But CG can often be used to resolve these ambiguities.

For example, in the Breton sentence:

(7) a. **D'ar mare ma oan o tigeriñ an nor**
b. *Au moment où j'étais en train d'ouvrir la porte* (*While I was opening the door*)

c. **Au moment mon agneau en train d'ouvrir la porte* (*At the moment my lamb in process of opening the door*)

both items in the collocation **ma oan** are ambiguous: **ma** can mean either *my* or *that*, and **oan** can mean either *lamb* or *[I] was*. The combination we want (*that* + *I was*) can be selected by using a rule:

```
LIST DetPos = (det pos);
LIST Vbloc = (vbloc);
LIST Vblex = (vblex);
LIST Vbser = (vbser);
LIST TempsFin = (pri) (pii) (past) (fti);
SET Verb = Vbloc | Vblex | Vbser;
SET VerbFin = Verb | TempsFin;
REMOVE DetPos IF (1 VerbFin);
```

which removes the possessive pronoun as a reading when a verb follows.

Todo: Have I missed an additional rule to disambiguate oan?

Another example from Welsh is the use of the inflected form of the preposition **i** (*to*) after a temporal adverb to create the subject of a subordinate clause:

(8) a. **cyn iddyn nhw fynd**

b. *before they go/went*

Here, **cyn** can mean either *as* or *before*, and **i** can mean either *to* or *I*. One way of dealing with this is to have two co-dependent rules. The first:

```
SELECT:S_cyn_cnjsub_prei ("cyn" cnjsub) (1 Prep_i);
```

uses the following preposition to decide that **cyn** is a subordinating conjunction *before* (the other meaning of **cyn**, the co-ordinating conjunction *as*, would always occur before an adjective). Then a further rule uses the disambiguation of cyn to deal with i:

```
SELECT:S_subj_postcyn PrnSubj (-1 ("cyn"ri cnjsub));
```

The subject pronoun reading is selected is the subordinating conjunction **cyn** is the previous word.

Todo: Perhaps omit the rule names here.

6 Disambiguating same-POS homonyms

In each homonym group above, the individual items differed from each other in terms of their part-of-speech information. However, Apertium can also use CG to disambiguate items in a same-POS homonym group. For example, in Welsh **cyfeiriad** can mean *address* or (less commonly) *direction*. In the sequence:

(9) a. **i drafod cyfeiriad y bolisi**

b. *to discuss the direction of the policy*

c. **to discuss the address of the policy*

we need to choose the less common meaning. In the Apertium Welsh dictionary, there are separate entries for both meanings, with the less common one given the differentiating tag *S1*:

```
<e><p>
<l>cyfeiriad<s n="n"/><s n="m"/></l>
<r>address<s n="n"/></r>
</p></e>

<e r="LR"><p>
<l>cyfeiriad<s n="n"/><s n="m"/><s n="S1"/></l>
<r>direction<s n="n"/></r>
</p></e>
```

A CG rule can then be used to select the appropriate meaning by substituting the *S1*-tagged entry for **cyfeiriad** when the verb **trafod** precedes it:

```
SUBSTITUTE ("cyfeiriad"ri n m) ("cyfeiriad" n m S1) NC
(-1 ("trafod"ri vblex));
```

7 Casting of POS tags

Moving on from the disambiguation of same-POS homonyms, we are exploring the implications of using CG to do pre-processing of polysemous lemmas (i.e. homonyms with related meanings) by adjusting their tags based on surrounding text. For instance, compare the two sentences:

(10) a. **Mi welsom ni y bachgen a'n ffrindiau ar y traeth.**
b. *We saw the boy and our friends on the beach*

(11) a. **Mi welodd y bachgen ni a'n ffrindiau ar y traeth.**
b. *The boy saw us and our friends on the beach*
c. **The boy saw we and our friends on the beach*

The lemma **ni** is polysemous between *we* (subject) and *us* (object). The entry in the Apertium Welsh dictionary is for the subject meaning only, so the second sentence will have the sub-optimal translation *we* instead of the desired *us*. It would be possible to add another dictionary entry for the object meaning, and use a CG rule to select it whenever **ni** is not preceded by a first-person plural verb, but a shortcut to the same result is to cast the subject tag attached to **ni** to an object tag, using a rule such as:

```
SUBSTITUTE (subj) (obj) prn_subj_1pl (NOT -1 vb_1pl);
```

Another alternative for dealing with this would be to remove the subject tag from the dictionary entry for **ni**, and then MAP a syntax tag onto it as required:

```
Rwyf/Bod<vbser><pres><p1><sg>
i/prpers<prn><p1><mf><sg><@←SUBJ>
```

This would involve changes to the bilingual dictionary and the transfer rules. The idea would be to map to subject tags first, and then use CG rules to map to object where necessary.

Todo: Can you say more about this? Should we weigh up pros and cons, or is it a bit pointless?

8 Porting of An Gramadóir rules for Irish

CG has already been used for Irish [2], with over 300 rules being reported. Given the relative lack of resources for marginalised languages, we were anxious to re-use work already done by Kevin Scannell on his Irish grammar-checker, and the first author has ported the disambiguation rules there to the **vislcg3** format.

Todo: Say more about why, how, and benefits.

9 Discussion

Although the integration of **vislcg3** into Apertium began as an ad hoc means of adding functionality, it has now become a much more important part of the Apertium pipeline. Currently, there are **xx**, **yy** and **zz** CG rules for Welsh, Breton and Irish respectively, and it is likely that the scope of the rules will expand as we become more adept at using them creatively.

It is worth noting that without the GPL licensing of **vislcg3** it would not have been possible to use it in Apertium. This would have meant coding something similar for Apertium, instead of spending that time improving the translation fidelity. In turn, this would have meant that the insights derived from applying CG to a couple of new languages would have been lost. GPL licensing, where both the software and the data are available for interested researchers to examine, has ensured optimal results for blah blah

Todo: This is crap - perhaps you can do better, or I'll look at it again tomorrow.

References

[1] FRANCIS M. TYERS AND KEVIN DONNELLY: *apertium-cy – a collaboratively-developed free RBMT system for Welsh to English*, Prague Bulletin of Mathematical Linguistics, 91, (2009), 57-66.

[2] ELAINE UÍ DHONNCHADHA AND J. VAN GENABITH: *A Part-of-speech tagger for Irish using Finite-State Morphology and Constraint Grammar Disambiguation*.

Todo: Need to add more references, eg Scannell, GPL papers. Some details are missing for Uí Dhonnchadha, and perhaps we should use a separate bibliography file.