Words: Building a morphological analyzer

Hakan Burden, 760128-4818

February 26, 2007

1 Introduction.

I have chosen to use SICStus Prolog for implementing the transducer and subsequently also the morphological description. The transducer implementation is in transduce.pl and the morphological description resides in inflections.pl.

The paradigms have been transferred into transitions for the transducer by hand. This might seem far to time-consuming and unsophisticated but I wanted to see how the transitions can be shared by the different paradigms.

I have also chosen to denote elements of the left language in the regular relation as tokens and elements of the right language as morphs. In a strict sense they are of course both symbols of there alphabets.

The following section describes the morphological description and the third section describes the transducer. Section four gives some examples of how to use the different predicates described in this report.

2 Morphological description.

The morphological description is implemented using the predicate transition/4, where the first argument position is for declinations. This is followed by an argument each for the start node and the end node of the transition. The last argument is the relation token:morph where token is in the language I={#, a, b, ..., z, ä, ö} and morph in 0=IU(+N1,...,+N7, +NEU,+UTR,+S2,+PL,+IND,+DEF,+NOM,+GEN).

A morphological symbol can be realized as a sequence of more than one character forcing the introduction of a special symbol for jump arcs. An example of the necessity for such arcs is the plural suffix ar. Jump arcs are also needed for the opposite case when a morph has no realization, like the singular morph. I have chosen to use # when there is no equivalent symbol in a relation. In order to make a morphological symbol like +N3 into a Prolog atom it is placed within single quotes, '+N3'.

Since a transition can be shared by a number of declinations the first argument is often substituted for the wild card, notated by _. An example of a shared transition is the follow-up suffixes for the singular forms of flicka and pojke when they have been labeled as belonging to the first and second declination respectively. In fact all declinations share the two possible ways of realizing the suffix for case. This is also true when a word ends on s, like ros and Tricks.

An analysis of a noun is on the form stem +Declination +Gender +Nerum +Defiteness +Case. Given the left sequence flickors the right sequence will be flicka +N1 +UTR +PL +IND +GEN where o is paired with +PL and r with #.
The seventh declension caused some problems for the plural-definite forms of utrum words. What is the proper realization of broiler +NOM +UTR +PL +DEF +NOM? Broilersma sounds strange to me and broilersen signals that the gender should be neutrum. In the end I settled for the latter. But it is an interesting issue how imported words are fitted into the existing inflectional patterns.

3 Implementing the transducer.

The main predicate of the transducer is transduce/4. The first argument is for the declension under discussion, the second for the current node and the two last for the sequence of tokens and morphs that have been so far acquired.

Transducing is a recursive procedure where we go from one state in the transducer to the next by a transition. These are given by the inflection patterns in inflections.pl.

As a consequence of jump arcs in the transitions there are four cases for transduce/4:

1. The base case. A relation is successfully read if we have no more symbols in either language to read and the current node is a final node.

2. The token of the left language has no equivalent morph on the right side.
   Only the third argument for acquired tokens will be updated after the transition.

3. The opposite situation from 2, with no equivalent token for a morph.

4. There is a one-to-one relation between the two languages in the relation.
   Both tapes are updated after the transition.

Furthermore the transducer can be run reading left-to-right or right-to-left giving the possibility to either obtain morphological representations from a surface level description or vice versa. In order to get all solutions to the two problems there are two driver predicates, one for generating, generate/1, and one for parsing, parse/1.

To use the transducer for solving another problem all you need is a suitable definition of final/1 and transition/4.

4 Examples.

\texttt{transduce/4:}

Given that the current node is 20 and we don’t care about declinations, what are the possible regular relations? The two possibilities are either #:+NOM or #:+GEN.

\texttt{| ?- transduce(_, 20, Tokens, Morphs). Morphs = [‘+NOM’]. Tokens = [] ? ; Morphs = [‘+GEN’]. Tokens = [s] ? ; no 2|}
parse/1:
What are the possible morphological analysis of the word *vals*? Apart from the five possibilities given in the assignment we also get the singular-indefinite-genitive cases for the second and third declinations.

```prolog
| ?- parse([v,a,l,s]).
val+N2+UTR+SG+IND+GEN
val+N6+NEU+PL+IND+GEN
val+N6+NEU+SG+IND+GEN
vals+N2+UTR+SG+IND+NOM
vals+N2+UTR+SG+IND+GEN
vals+N3+UTR+SG+IND+NOM
vals+N3+UTR+SG+IND+GEN
yes
```

generate/1:
The genitive *s* is dropped if the last token in the word is *s*.

```prolog
| ?- generate([r,o,s,'N1','+UTR','+SG','+IND','+GEN']).
ros
yes
```