2

Inducing a Morphological Transducer from Inflectional Paradigms

LAURI CARLSON

A traditional way to represent the morphology of inflectional languages is through *paradigms*. This paper presents an idea and a program to induce a nondeterministic morphological transducer from traditional style paradigm sets.

2.1 Paradigm morphology

A traditional way to represent the morphology of inflectional languages is through *paradigms*. A paradigm (the Greek word for example) is a list or table of inflectional forms of an example word, representing a given inflectional class. The table is indexed by grammatical tags, and the items in the table cells are inflected forms. In linguistic morphology, this approach is known as the WP (Word and Paradigm) model.

Ideally, to find a given form of a new member of the same class, one substitutes the inflectional stem of the new word in place of the stem of the paradigm word and reads off the resulting form. In grammars intended for human consumption, the relation between the paradigm and its representatives may be subject to simple morphophonological rules sometimes left for the human user to figure out.

Compared to concatenative (IA, Item and Arrangement) morphology, the WP model differs by just listing the forms, without breaking them up into correspondences between individual tags and morphs. For instance, the paradigm of Latin singular nominative noun like *servus* only tells that the genitive plural is *servorum*. Compared to rule (IP, Item and Process) morphology, the corre-

Inquiries into Words, Constraints and Contexts.
Antti Arppe et al. (Eds.)

spondences do not go down to segment level, so there is no explicit treatment of morphophonology.

Paradigms in a WP morphology can be identified by arbitrary labels (declension or conjugation number) or by a set of thematic forms which suffice to identify the paradigm. For instance, Latin verb *amo* belongs to first conjugation, identified by the thematic form series *amo, amavi, amatum, amare*.

2.2 Inflectional morphology

`murf` is a small Prolog program intended to induce from traditional style paradigm sets a morphological transducer which (1) produces the forms in the paradigms, (2) does not produce any forms either explicitly or implicitly excluded from the paradigms, and (3) generalises common features of the paradigms, reducing redundancy in the paradigms.

The initial idea was quite simple. `murf` reads in forms in a set of tagged forms, trying to place each form in a two-tape finite state network, maximising the match of the new form in the existing network. The new form is matched with the existing network at both ends of the net. A match which leaves the least unmatched residue is chosen, and the missing part is added into the net as a new arc.

Given, for instance, a paradigm

```
Form           Tagging
talossa         talo 1 N SG INE
taloissa        talo 1 N SG INE
talona          talo 1 N SG ESS
```

`murf` correctly infers that the plural essive form is `taloina`:

```
0: talo talo 1 N 4
 4  SG 5
  5 ssa INE 1.
  5 na ESS 1.
 4 i PL 5
5
```

(The number following the base form identifies the base as a member of a given paradigm. The numbering follows that of *Nykysuomen sanakirja* (Dictionary of Contemporary Finnish). As the net shows, `murf` is able to infer a segmentation of the forms into morphs and tags the morphs appropriately. As a side effect of entering the attested form in the network, new, unattested forms may get generated through re-entrances in the net. Call such forms side effects.

The initial idea has gone through a number of refinements to capture famil-

iar morphological phenomena in real data. They include *morphotax, complementary distribution, free variation, blocking, defective paradigms* and *productivity*.

2.2.1 Morphotax

Morphotax concerns the admissible orders of tags in a well-formed word. The heuristics `murf` follows here is that a proposed match of a new word is not allowed to produce unattested taggings.

To guarantee that, `murf` first forms a separate one-tape morphotax network of the taggings it has encountered. When a new form is considered for entry at a given place of the net, its side effects are first checked for morphotax.

2.2.2 Complementary distribution

Complementary distribution is present when any given tagging is realised by just one form, although tags occurring in it have more than one allomorph. For instance, Finnish partitive endings `tA` and `A` are in complementary distribution, the former occurs after heavy syllables and the latter after light ones. Identically tagged forms are *free variants*. `murf` implements complementary distribution by preventing production of free variants as a side effect of insertion.

2.2.3 Free variation

To allow genuine free variation to get past the complementary distribution check, it suffices to tag the variants as different. For instance, Finnish third person possessive suffix has two forms `nsA` and `Vn` which are in free variation after light open syllables. They are tagged as P3/A and P3, respectively. If desired, the distinguishing tags can be merged afterward.

2.2.4 Blocking

Blocking refers to the phenomenon that a lexicalised exception to a regular rule blocks a productive, regular rule. For instance, Finnish nominative plural is `talot`, not `taloi`, as one might be led to expect from the previous data. `murf` accounts for blocking in the following way. When a paradigm is read in, all forms in it are put on a waiting list. Whenever a form is inserted, forms on the waiting list are checked for blocking. An insertion is not allowed if it would produce a side effect blocked by a form on the waiting list.

2.2.5 Defective paradigms

Some paradigms are *defective* in that some forms are missing from an expected cross classification. For instance, Finnish comitative and instructive (instrumental) cases only have one number (plural). From a combinatorial point of view, case and number form in these cases a *portmanteau* morph

instead of two independent morphs. A straightforward way of recording this gap in distribution is to make the tag combination `PL_COM` a tag on its own. Again, the tag can be normalised afterward if needed..

Another distributional gap is that nouns do not occur in comitative plural without possessive suffix (adjectives do). To record such gaps `murf` allows defining, alongside the networks of legitimate forms, separate networks for exceptions. For instance, entry

```
*-      - N PL_COM
```

disallows nouns ending in plural comitative.

2.2.6 Productivity

Productivity refers to the fact that certain forms generalise by default to new words, while others are restricted to a closed set of forms. (This fact is one of the main motivations of paradigm morphology in the first place.) For instance, Finnish nominals have productive vowel stems and less productive consonant stems. A new base form like *pokemon* will automatically go in the productive wovel stem paradigm. `murf` allows marking a variant as a nonproductive one as follows:

```
tienoisiin     tienoo 24 N PL ILL
tienoihin       tienoo 24 N PL_ILL/h!
tienoiden       tienoo 24 N PL GEN
tienoitten      tienoo 24 N PL_GEN/tt!
```

Nonproductive variants marked with `!` will not be generalised into paradigms where they have not been specifically licensed by attested forms.

2.3 Derivational morphology

Derivational morphology allows concatenating base forms coming from different paradigms. A derivational affix may be specific (at least) to part of speech. For instance, Finnish abessive adjective suffix `tOn` produces an adjective out of a noun.

`murf` allows constraining derivational endings with a categorial grammar style tag format `X\Y`

```
onneton onni 8 N tOn N\A 57 A SG NOM
```

This constrains `tOn` to combine with nouns and produce adjectives. (Formally, `X\Y` is analogous to a portmanteau tag discussed above in that it constrains variation at a point in the net.)

2.4 Theory and implementation

Roughly, the idea behind `murf` is to look for a minimal nondeterministic acyclic transducer which produces all of the forms on a list of legitimate

tagged word forms, none of the forms on a list of illicit forms, and produces each legitimate tagging just once. (Free variation is handled by tagging free variants as different and erasing the differences afterward.) The size of the machine is measured by the numbers of states and arcs and the length of the labels on the arcs.

Finding a minimal nondeterministic network for a list of forms is a NP complete problem, Tamm (2004). Furthermore, it is not likely to have a unique solution. There may be more than one minimal machine for any given set of forms, with different side effects.

As stated, the problem is independent of the order of the items on the list(s). `murf`, on the other hand, is order dependent. Its task is to find for each form on the list the smallest extension of the network built so far which produces at least that form, under the same constraints as before. The algorithm is roughly this:

- take a form from list
- while possible
  - find a new minimal arc which generates the form in the network
  - check the arc against constraints on list and net
  - compare the arc to the best find so far
- insert the best arc in network

In `murf`, "smallest extension" is measured in terms of the length of the infix arc. `murf` now actually goes through all minimal infixes which produce the string, testing them against the list and against the network for duplicates, and choosing the shortest among them. This can take long, because there may be many minimal infixes, and each infix may produce many forms which must be compared to the list as well as to the net for duplicates. The minimal infixes depend on the form, so they are computed again for each form.

2.5 Experiments

`murf` has been tested with Finnish nominal and verb paradigms. There are ca 80 nominal and 50 verb paradigms respectively, in the classification of the Dictionary of Contemporary Finnish (original edition). In an initial test, a set of 80 noun paradigms producing around 20,000 forms got coded into a nondeterministic transducer with around 600 states and 1,700 arcs, almost 1,000 of which were epsilon arcs. Only correct forms get produced.

The largest run so far took 34 hours wallclock time. It loaded a training set of about 5,000 word forms to produce a nondeterministic network for about 140 word paradigms, among them all examples of the nominal inflection paradigms listed at front of the Dictionary of Contemporary Finnish, including rare and obsolete patterns. A total of 463,282 word forms get generated from a network of 564 states and 1,647 arcs.

The size of the C runtime `mph` is about 90 kilobytes and runs through the forms in about 4.3 seconds, more than 100 words per millisecond. It stores about 100 words per state, about 5 words per byte.

With little space/time optimisation done in `murf` so far, adding new paradigms gets slow toward the end of the process. There are likely to be ways to make `murf` more time efficient by caching or reordering tasks.

Many forms appear to produce the same illegal side effects. It seems to pay to cache recurrent illegal forms: in a recent test run, the run time went down three fourths as a result.

2.6 Adding new forms to existing paradigms

Adding new words to existing paradigms can be made faster once the paradigm is in. The original idea of `murf` was precisely to implement this insight of the WP model. Thanks to the organisation of the network, new words only need to be given a few thematic forms for the word to settle in the right places in the network, and produce the predictable forms as side effects.

2.7 Guessing forms

It is also possible to use the net to guess the paradigms of unknown words on the basis of thematic forms.

2.8 A runtime morphology analyser/generator in C

The network defined by `murf` can be dumped in the form of a C language string lookup table indexed by a one character lookahead. A tiny (250 lines, 50K) C runtime parser/generator `mph` does lookup from the table at the rate of 100 words per millisecond (raw listing).

2.9 Order sensitivity

murf is sensitive to the order in which forms are presented to it. If regular paradigms are presented before irregular ones, murf tends to overgeneralise, and subregularities across irregular paradigms may get missed. The best strategy seems to be to start with paradigms which exhibit central regularities but make significant splits between regular and irregular sets of endings. In Finnish nouns, a good strategy proved to be to start with bisyllabic nouns in paradigm 40 (`susi 'wolf', vesi 'water'`) whose local cases are regular, while irregular grammatical cases show stem allomorphy.

2.10 Discussion

There are by now a variety of approaches learning morphology from data. Koskenniemi (1991) considers learning two-level morphophonological rules

from surface alternations. Goldsmith (2000) presents a heuristics and a statistical evaluation procedure to find a morphological segmentation for a language from a raw text corpus. Creutz et al. (2005) carry this idea further and make an unsupervised morphological segmenter available for download.

`murf`, in contrast, belongs to the paradigm of supervised learning, as it expects fully tagged and classified paradigms painstakingly prepared and sorted by a linguist, restricting itself to the task of converting the paradigms into a less redundant form. From a linguistic point of view, murf can be seen to implement some of the traditional principles of taxonomical morphemic analysis.

As a solution of the theoretical minimisation problem, `murf` remains naive. More robust methods could be found to optimise the transducer induction task as a purely computational problem. As they are, `murf` and `mph` may just about do for generating small scale morphological analysers and generators for restricted natural language tasks.

References

Creutz, Mathias, Krista Lagus, Krister Linden, and Sami Virpioja. 2005. Morfessor and hutmegs: Unsupervised morpheme segmentation for highly-inflecting and compounding languages. In *Proceedings of the Second Baltic Conference on Human Language Technologies*, pages 107–112. Tallinn.

Goldsmith, Jerry. 2000. *Unsupervised Learning of the Morphology of a Natural Language*. University of Chicago.

Koskenniemi, Kimmo M. 1991. A discovery procedure for two-level phonology. In L. Cignoni and C. Peters, eds., *Computational Lexicology and Lexicography: A Special Issue Dedicated to Bernard Quemada*, pages 451–446.

Tamm, Hellis. 2004. *On Minimality and Size Reduction of One-Tape and Multitape Finite Automata*. Department of Computer Science Series of Publications A,Report A-2004-9. Helsinki: University of Helsinki Department of Computer Science.