A Multilayered Declarative Approach to Cope with Morphotactics and Allomorphy in Derivational Morphology

Johannes Handl, Carsten Weber

Friedrich-Alexander-Universität Erlangen-Nürnberg
Professur für Computerlinguistik
Bismarckstr. 6, 91054 Erlangen
jshandl@linguistik.uni-erlangen.de, cnweber@linguistik.uni-erlangen.de

Abstract

This paper deals with the derivational morphology of automatic word form recognition. It presents a set of declarative rules which augment lexical entries with information governing the allomorphic changes of derivation in addition to the existing allomorphy rules for inflection. The resulting component generates a single lexicon for derivational and inflectional allomorphy from an elementary base-form lexicon. Thereby our focus lies both on avoiding redundant allomorph entries and on the suitability of the resulting lexical entries for morphological analysis. We prove the usability of our approach by using the generated allomorphs as the lexicon for automatic wordform recognition.

1. Introduction

One of the main concerns when coding a grammar for a natural language morphology is to provide the information needed for dealing with its allomorphic phenomena in a detailed and coherent manner. Independent of the chosen theoretical background, be it a Two-Level (Koskenniemi, 1983; Trost, 1990; Trost, 1993), a paradigmatic (Calder, 1989), an object-oriented (Daelemans et al., 1992; Evans and Gazdar, 1996; Riehemann, 2000), or an LA (Hausser, 1992; Hausser, 2006; Hausser, 2009) approach, the grammar developer is confronted with the problem that in most languages allomorphy emerges as the result of different varieties of word formation.

A well-known phenomenon of German allomorphy is a vowel mutation, the so-called 'Umlautung'.[1] Many German nouns use an additional allomorph to form their plural. E.g., the noun *Buch*, the German pendant for *book*, has two inflectional stem allomorphs, the singular stem allomorph *Buch* and the plural stem allomorph *Büch*. Allomorphy, however, may also be triggered by a derivational affix like the suffixes *chen* or *lein* which are usually used to create a diminutive form, e.g., the diminutive form of *Buch* is *Büchlein*.

It is evident that an allomorphic inflectional form is neither a sufficient nor a necessary condition for an allomorphic derivational form. (⇒) The plural stem of the German noun *Fach* is *Fäch*, but the derivational adjective is *fachlich*. (⇐) The singular and the plural stem of the German noun for flower, *Blume*, are the same, whereas the diminutive form *Blümlein* is the result of a vowel mutation and of an e-elision. We conclude that allomorphy phenomena of inflection and of derivation have to be treated alike, but independently of each other, cf. Trost (1990).

The definition for the derivational morphotactics must not only specify the preconditions, but also provide the necessary information to choose from a subset of quasi equivalent suffixes. For example, even though there are predominantly phonetical rules to choose between the suffixes *chen* or *lein* in German, these rules often allow more than one alternative. In such a case, the choice is often used to modify the semantics of the formed expression. For example, the diminutive form *Frauchen* refers the owner of a pet, usually a dog, whereas the diminutive form *Fräulein* is the old-fashioned and now political incorrect denomination of a rather young and unmarried woman.

Because some stem allomorphs are required for derivation only, it is necessary to investigate the exact set of combinable derivational affixes for a given stem. Determining the required allomorphs and their morphotactics requires a lot of work if it is done manually.[2]

This raises the question whether the morphotactics of a single derivational affix should be determined separately or whether it is advisable to treat groups of affixes as a whole. The answer depends on the language being described. In German, for example, there exists a certain correlation between the allomorph stems and the compatible derivational affixes. Thereby, several derivational rules resulting in 'Umlautung' or e-elision share some of the preconditions for the allomorphic change; however, the scope of grouping is limited, as some of the preconditions may differ – as in the case of the suffixes *lich* vs. *isch* in *handlich* vs. *händisch*.

From a technical point of view it seems easiest to define the allomorphs resulting from inflection and derivation independently from each other. As a result, inflection can be handled paradigm-based, while the derivational component may be switched on and off, dependent on the application. This, however, requires a mechanism to merge the allomorphs for inflection and derivation while avoiding unnecessary ambiguities resulting from redundant generation.

[1] For a concise introduction to the topic of word formation in German morphology, cf. Fleischer and Barz (1992), Motsch (2004).

[2] The task can be simplified by using large corpora, though, admittedly, postprocessing is still needed as derivation is rather productive in a number of languages. Besides, many derivational forms a rarely used and, consequently, many derivational forms will be missing even in very large corpora.

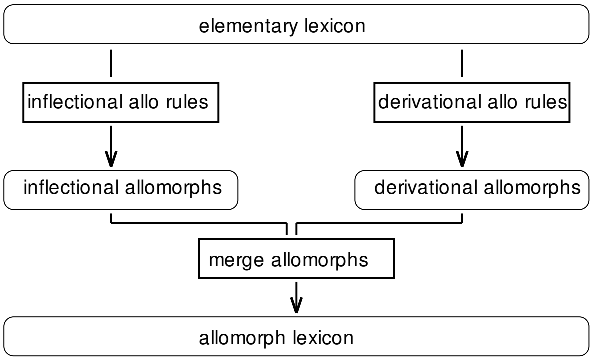

Figure 1: Flow chart of allomorph generation

2. Idea

An adequate handling of derivation should guarantee that a given derivational affix can only be combined with the appropriate stem allomorphs and only with these, thus avoiding under- and over-generation.[3] It is also desirable that the automatically generated allomorph lexicon does not contain any redundant allomorph entries. This goal can be achieved by a rule-based approach in the form of several cascaded preprocessor steps which refine a given lexicon.

Our procedure is in accordance with the allomorph method presented in Hausser (1999). Based on an elementary lexicon, an allomorph lexicon is created before runtime serving as the backbone of a morphological analysis during runtime. However, we improve this approach by inserting both a further preprocessor step to generate the derivational allomorphs and a subsequent merging step which fuses the resulting allomorphs of the precedent steps.

The complete generation process can be seen in fig.1. The inflectional as well as the derivational allo rules operate on the same elementary lexicon in order to generate the required allomorphs for inflection, the *inflectional allomorphs*, and the necessary allomorphs for derivation, the *derivational allomorphs*, which together form the allomorph lexicon. Conflating the obtained allomorphs is essential, as inflectional and derivational allo rules may generate the same allomorph more than once. Note that the derivational allomorph rules alone may already generate redundant allomorphs.

Although it is technically feasible to apply the inflectional and the derivational allo rules to the same elementary lexicon (as described above), it may be a good idea to split the elementary lexicon in two parts, one for the simple and one for the derivative word forms. Storing the base-form entries for the generation of the inflectional and derivational allomorphs apart renders the compilation of the lexicon files much easier, as enriching a single lexicon with the complete information regarding any mechanism of word formation would breach the structuring of the lexicon which hitherto merely reflects the inflectional paradigms.

3. Implementation

Technically, the implementation builds on the JSLIM allomorph preprocessor (Handl et al., 2009), which enables the user to define the way allomorphs are created in a declarative manner.[4]

The preprocessor interprets allo rules which specify the surface and the category values of the generated allomorphs. It supports flat non-recursive feature structures as the underlying data structure of the lexicon entries. Besides, *templates* can be used to avoid redundancy by sharing common attribute-value pairs as can be seen in fig.2, 3 and yet another kind of rules, so-called *combi rules*, are used during analysis to describe the combination with inflectional affixes for a given flectional paradigm.

Fig.2 shows an extract from the elementary lexicon. It defines a template which comprises two attributes, the attribute `allo` and the attribute `combi` with the value `A_Frau` and `C_Gabe` respectively. Subsequent to the template are the associated entries, which provide possible values for its attribute `sur`.

```
!template[allo: A_Frau, combi: C_Gabe]
![sur]

A-Dur
A-Moll
ABM-Stelle
Abhitze
Abiose
Ablepsie
...
Blume
...
```

Figure 2: Extract from the elementary lexicon

```
!template[allo: A_chen]
![sur]
Amboss
Anekdote
Backe
Bahn
Balken
Balkon
...
Blume
...
```

Figure 3: Extract from the derivational elementary lexicon

The lexicon entries for the generation of the allomorphic variants of the derivative forms are either merged in the preprocessor step or combined with a derivational suffix during analysis. Hence, we do not have to assign a combi table. However, as illustrated in fig.3, an allo rule is required to describe the necessary allomorphic changes.

We store the information about the valid combination of an allomorph stem with a derivational affix by means of an additional attribute, here via the attribute *der*. The values of this attribute are the surfaces of the derivational affixes with which the allomorph stem can be combined.

[3] It would be useful to deploy semantic knowledge to constrain the legitimate combinations even more, but this is not the topic of this paper.

[4] As a consequence the output is generated in a form which is suitable for further processing with JSLIM and can be directly used as the allomorph lexicon for a morphology written in JSLIM.

$$\begin{bmatrix} \text{sur:} & \text{chen} \\ \text{core:} & \text{chen} \\ \text{dern:} & \text{C\_Maedchen} \\ \text{sfxn:} & \text{(n)} \\ & \\ & \\ & \\ & \\ & \\ & \\ & \\ & \\ & \end{bmatrix} \begin{bmatrix} \text{sur:} & \text{er} \\ \text{noun:} & \text{er} \\ \text{cat:} & \text{(s3)} \\ \text{sem:} & \text{(perspro)} \\ \text{allo:} & \text{A\_achteln} \\ \text{combi:} & \text{C\_null} \\ \text{dern:} & \text{C\_Adler} \\ \text{derv:} & \text{C\_achteln} \\ \text{flxs:} & \text{(nva)} \\ \text{pfx:} & \text{(v)} \\ \text{sfxn:} & \text{(nv)} \\ \text{sfxv:} & \text{(nv)} \\ \text{gra:} & \text{fug} \end{bmatrix}$$

Figure 4: Coding of an derivational affix

Fig.4 shows the entry for the derivational affix *chen*. Apart from its surface and core value, the entry comprises the two attributes `dern` and `sfxn` to mark it as a derivational suffix for nouns. Thereby, `dern` contains the inflectional information of the resulting derivative word form whereas `sfxn` denotes the resulting part of speech (noun). The notation used also allows the coding of suffixes which may be combined with different kind of stems or coincide with other word forms as can be seen at the example of the suffix *er*. The latter can be combined both with nouns (as in *Spiel* → *Spiel-er* or *Wild* → *wild-er-n*) and with verbs (as in *schwimmen* → *Schwimm-er* or *schlafen* → *schläf-er-n*) both to build noun or verb forms. The respective inflectional classes are stored in the attributes `dern` and `derv`. The values of the attributes `sfxn` and `sfxv` denote the indefiniteness of the resulting parts of speech. Note, that *er* can also be used as a prefix, a personal pronoun or an epenthesis.

$$\begin{bmatrix} \text{noun:} & \text{Blume} \\ \text{allo:} & \text{F\_Blume} \end{bmatrix} \begin{bmatrix} \text{noun:} & \text{Mann} \\ \text{allo:} & \text{F\_Mann} \end{bmatrix}$$

```
table F_Blume: [noun] => [sur,noun,cat,sem]
  /(.*)/ => /$0/ /$0/ (f) (f sg)   .

table F_Mann: [sur] => [sur,noun,cat,sem]
  /(.*)([aou])([^aeiou]+)/ => /$0/ /$0/ (m-g) (m sg) ;
                           => /$1"$2$3/ /$0/ (stem) (m pl) .

table D_chen: [sur] => [sur,noun,der]
  /(.*)([uou])([^aeiou]*e)/ => /$1"$2$3/ /$0/ (chen) .

table D_lich: [noun] => [sur,noun,der]
  /(.*)([uou])([^aeiou]*e)/ => /$1"$2$3/ /$0/ (lich) .
                   /(.*)e/ => /$1/      /$0/ (lich) .
                  /(.*)en/ => /$1ent/   /$0/ (lich) .
```

$$\begin{bmatrix} \text{sur:} & \text{Blume} \\ \text{noun:} & \text{Blume} \\ \text{cat:} & \text{(f)} \\ \text{sem:} & \text{(f sg)} \\ & \end{bmatrix} \begin{bmatrix} \text{sur:} & \text{Blüm} \\ \text{cnoun:} & \text{Blume} \\ \text{der:} & \text{(chen)} \\ & \\ & \end{bmatrix} \begin{bmatrix} \text{sur:} & \text{Mann} \\ \text{noun:} & \text{Mann} \\ \text{cat:} & \text{(m-g)} \\ \text{sem:} & \text{(m sg)} \\ & \end{bmatrix} \begin{bmatrix} \text{sur:} & \text{Männ} \\ \text{noun:} & \text{Mann} \\ \text{cat:} & \text{(stem)} \\ \text{sem:} & \text{(m pl)} \\ \text{der:} & \text{(chen lich)} \end{bmatrix}$$

Figure 5: Elementary lexicon, allo rules and their result

A sketch of the application of the allo rules for the German nouns *Blume* and *Mann* and the derivational suffixes *chen* and *lich* can be seen in fig.5. The definition of an allo rule starts with the keyword *table* followed by its name and its signature. The latter defines which attributes are checked and which attributes are altered.

The body of the allo rules comprises one or more clauses separated by a semicolon or full stop which specifies the exact values of the precondition (left side) and postcondition (right side). The difference between using a full stop and a semicolon is that the former allows to inherit the missing values from the preceding clause. Regular expressions are marked by a pair of slashes. The JSLIM allo preprocessor supports the standard set of regular expressions and an additional operator, ", to realize vowel mutations.

Obviously, the allo rules D_lich, D_chen and F_Mann all generate the allomorph *Männ*. The fact that the allomorphs generated in this way are redundant, can be realized by observing their surface and the value of their core attribute. In the merging step, they are fused into a single allomorph the attributes of which are the result of the concatenation of their respective values. The order of the concatenation thereby depends on the order of the allo rules.

Note that it is essential that the part of speech is already stored in the elementary lexicon as the derivational affixes may be restricted in this respect.

4. Evaluation

The set of allomorphs generated in this way was evaluated by means of a German morphology (Handl et al., 2009), which was built using JSLIM and, up to now, did not allow a rule-based handling of derivation. The original morphology grammar used a base-form lexicon which comprised about 62.600 base-forms of nouns, 17.400 base-forms of adjectives, and 12.000 base-forms of verbs. In order to cope with the allomorphic changes of frequent derivative word forms, the original morphology grammar required additional lexicon files, which contained these allomorphic variants and comprised 7.900 derivative nouns, 1.200 derivative adjectives and 1.000 derivative verbs.

By adding rule base support for both derivation and composition, we were able to reduce the number of base-form entries for nouns, adjectives and verbs by 56.7%, 42.4%, and 51.4% respectively, thus obtaining a morpheme lexicon, which comprised merely 27.000 entries for nouns, 10.000 entries for adjectives, and 5.300 entries for verbs. These numbers could still be decreased by extending our rule system, e.g., to handle derivative word forms and compounds which contain hyphens, so that the corresponding base-forms could be removed from the lexicon.

	nouns	verbs	adj.	all
simple forms	28545	10565	6777	45887
derivative forms	10393	907	1194	12494
total	38938	11472	7971	58381
merged	28387	10557	6771	45715
reduction rate	27.1%	8.0%	15.1%	21.7%

Table 1: Required entries in the allomorph lexicon

As a direct consequence of merging the 12.500 allomorphic variants of the derivative forms with those of the simple forms as presented in this paper, most of the former could be spared as they overlapped with existing entries. So, instead of the initial 10.300 entries for derivative nouns only 158 entries were required. This accounts to a reduction of 98.5%. The number of required entries for adjective and verb derivatives also decreased by 99.1% and 99.5% respectively. With regard to the total lexicon size of 58.300 allomorphs this still amounts to an additional reduction of 21.7% (see tab.1).

Evidently, using a supplementary preprocessor step to fuse the inflectional and derivational allomorphs is worthwhile. Apart from the obvious benefit concerning the required memory, it also decreases the number of ambiguities perceptible by about 10-30%[5] as artificial ambiguities due to redundant lexicon entries could be avoided.

Although the here presented technique may have a slighter effect on other languages depending on the respective significance and variation of derivation, it does not restrict itself to a specific language and can therefore easily be adopted to any European language. Moreover, its generality renders it independent of a specific formalism so that it may be used regardless of the pursued theoretical approach.

5. Conclusion

In this article, we have shown that by using a declarative rule scheme and a multilayered allomorph generation method, an inflectional morphology can easily be extended to cope with derivational allomorphy.

Obviously, it is also possible to extend the approach even further and add another layer for the generation of allomorphs based on composition, though this might not be expedient for all languages. Likewise, this additional preprocessor step can be employed to facilitate further tasks concerning the mechanisms of word formation, e.g., the coding of epenthesis.

Our evaluation, which based on a computational morphology for German, showed that the size of the allomorph lexicon used for analysis could be reduced significantly, though there is still room for further improvements. Moreover, by splitting the elementary lexicon in two parts, one for the generation of the allomorphs which are required for the simple forms, and one for the generation of the allomorphs, which are required for the derivational forms, we managed to preserve the paradigm based structuring of the lexicon files and so facilitated enormously the compilation of the lexicon files.

6. Acknowledgment

We want to express our thanks to the members of the German morphology team of the University of Erlangen-Nuremberg, Paul Greiner, Sylvia Jaegers, Sylvia Kosche, Stefan Reihl, and Theresia Schneider, whose comments and remarks have been extraordinary helpful and who helped with the practical examination. Potential nonconformities are those of the authors alone.

7. References

Jonathan Calder. 1989. Paradigmatic morphology. In *Proceedings of the fourth conference on European chapter of the Association for Computational Linguistics*, pages 58–65, Morristown, NJ, USA. Association for Computational Linguistics.

Walter Daelemans, Koenraad De Smedt, and Gerald Gazdar. 1992. Inheritance in natural language processing. *Comput. Linguist.*, 18(2):205–218.

Georgette Dal, Nabil Hathout, and Fiammetta Namer. 1999. Construier un lexique dérivationnel: théorie et réalisations. *Conférence TALN.*

Roger Evans and Gerald Gazdar. 1996. DATR: a language for lexical knowledge representation. *Comput. Linguist.*, 22(2):167–216.

Wolfgang Fleischer and Irmhild Barz. 1992. *Wortbildung der deutschen Gegenwartssprache*. Max Niemeyer Verlag.

Johannes Handl, Besim Kabashi, Thomas Proisl, and Carsten Weber. 2009. *JSLIM – Computational Morphology in the Framework of the SLIM Theory of language*. Springer.

Roland Hausser. 1992. Complexity in left-associative grammar. *Theoretical Computer Science*, 106(2):283–308.

Roland Hausser. 1999. *Foundations of Computational Linguistics*. Springer, Berlin and Heidelberg.

Roland Hausser. 2006. *A Computational Model of Natural Language Communication: Interpretation, Inference, and Production in Database Semantics*. Springer, Berlin and Heidelberg.

Roland Hausser. 2009. Modeling natural language communication in database semantics. *Proceedings of the APCCM, Australian Computer Science Inc. CIPRIT, Vol. 96.*

Kimmo Koskenniemi. 1983. *Two-level Morphology: A General Computational Model for Word-Form Recognition and Production*. Ph.D. thesis, University of Helsinki.

Wolfgang Motsch. 2004. *Deutsche Wortbildung in Grundzügen*. de Gruyter.

Fiammetta Namer. 1999. Le traitement automatique des mots dérivés: le cas des noms et adjectifs en -et(te). *Colloque "La morphologie des dérivés évaluatifs"*.

Fiammetta Namer. 2000. FLEMM: Un analyseur flexionnel du français à base de règles. *Traitement Automatique des Langues (TAL)*, 41(2):523–547.

Susanne Z. Riehemann. 2000. Type-based derivational morphology. *JOURNAL OF COMPARATIVE GERMANIC LINGUISTICS*, 2:49–77.

Harald Trost. 1990. The application of two-level morphology to non-concatenative german morphology. Research Report RR-90-15, DFKI, Saarbrücken.

Harald Trost. 1993. Coping with derivation in a morphological component. In *In Proceedings of 6th EACL*, pages 368–376.

[5] This number not only depends largely on the average ambiguity rate of the computational morphology, but also on the chosen test set and is therefore little meaningful, except for showing that there is an improvement.