A Rule-based Morphological Analyzer for Murrinh-Patha

Melanie Seiss

University of Konstanz
Konstanz, Germany
melanie.seiss@uni-konstanz.de

Abstract

Resource development mainly focuses on well-described languages with a large amount of speakers. However, smaller languages may also profit from language resources which can then be used in applications such as electronic dictionaries or computer-assisted language learning materials. The development of resources for such languages may face various challenges. Often, not enough data is available for a successful statistical approach and the methods developed for other languages may not be suitable for this specific language. This paper presents a morphological analyzer for Murrinh-Patha, a polysynthetic language spoken in the Northern Territory of Australia. While nouns in Murrinh-Patha only show minimal inflection, verbs in this language are very complex. The complexity makes it very difficult if not impossible to handle data in Murrinh-Patha with statistical, surface-oriented methods. I therefore present a rule-based morphological analyzer built in XFST and LEXC (Beesley and Karttunen, 2003) which can handle the inflection on nouns and adjectives as well as the complexities of the Murrinh-Patha verb.

Keywords: Finite-state morphological analyzer, under-resourced languages, morphologically rich languages

1. Introduction

Resource development mainly focuses on well-described languages with a large amount of speakers. However, smaller languages may also profit from language resources, which can then be used in applications such as electronic dictionaries or computer-assisted language learning materials. The development of resources for such languages may face various challenges. Often, not enough data is available for a successful statistical approach and the methods developed for other languages may not be suitable for this specific language.

In this paper, I present a morphological analyzer for Murrinh-Patha, a polysynthetic language spoken in the Northern Territory of Australia. Murrinh-Patha currently has around 2500 speakers, but the speech community is healthy and expanding, and is thus not considered endangered by Australian standards. While nouns in Murrinh-Patha only show minimal inflection, verbs in this language are very complex. This complexity makes it very difficult if not impossible to handle data in Murrinh-Patha with statistical, surface-oriented methods. I therefore present a rule-based morphological analyzer built in XFST and LEXC (Beesley and Karttunen, 2003), which can handle the inflection on nouns and adjectives as well as the complexities of the Murrinh-Patha verb.

The paper is structured as follows. First, section 2 presents related work. Then, section 3 describes some sample complexities of the Murrinh-Patha verb to show what challenges a morphological analyzer for Murrinh-Patha has to face and why a statistical approach is very difficult for this language. Section 4 deals with the implementation for the Murrinh-Patha verb. It shows how the rule-based methods can be used to model the details of the complexities as closely as possible.

How the implementation can be put to use in a successful morphological analyzer is the topic of section 5. It describes how a refined lookup strategy is used for a successful morphological analyzer, i.e., one with large coverage and high precision at the same time. Section 6 then evaluates the approach described in section 4 on a small Bible corpus. Section 7 concludes the paper.

2. Related work

Since Koskenniemi (1983) introduced the two-level approach to computational morphology and applied the formalism to Finnish morphology, finite state morphologies have been developed for a number of diverse languages, language families and language types. Some examples are the treatment of South Asian languages such as Bögel et al. (2007) for Urdu or Veerappan et al. (2011) for Kannada, the treatment of Indonesian by Larasati et al. (2011) as a representative for Austronesian languages or for Estonian (Uibo, 2005) and Finnish (e.g. Lindén and Pirinen (2009)) from the Uralic language family.

Semitic languages have traditionally been in the focus of finite-state methods for the treatment of their morphology due to the special challenges the root-and-pattern morphology poses. Some examples of treatments of the Semitic languages are, e.g., Beesley (1996) and Attia et al. (2011) for Arabic as well as Yona and Wintner (2008) for Hebrew. While all these approaches face different challenges posed by the respective morphology of the language, to my knowledge so far no finite state implementation of an Australian language with such a complex templatic verbal morphology as is found in Murrinh-Patha has been proposed. Closest to the implementation of Murrinh-Patha described here is probably the treatment of Basque by Alegria et al. (1996) and the treatment of Persian by Megerdoomian (2004).

The implementation of the Persian morphology proposed by Megerdoomian (2004) is concerned with problems involving tokenization, phonological rules and long-distance dependencies. However, it seems that the long-distance de-

pendencies for Persian, which are modeled with flag diacritics of the U-type, are simpler than the ones found in Murrinh-Patha.

Alegria et al. (1996) in their implementation for Basque propose a formalism for long-distance dependencies, but treat only simple long-distance dependencies. Their implementation is, however, relevant because it proposes a 3 step lookup strategy similarly to the lookup strategy used in this paper. The lookup strategy for Basque involves a standard analyzer, an analyzer of linguistic variants such as dialectal uses and a guesser.

Most recent projects either use XFST (Beesley and Karttunen, 2003), or the open-source alternatives Foma (Hulden, 2009) or HFST (Lindén et al., 2011). XFST is, for example, used in the implementations by Yona and Wintner (2008) for Hebrew or by Bögel et al. (2007) for Urdu. On the other hand, Attia et al. (2011) and Larasati et al. (2011), for example, explicitly state that they use Foma because of licensing issues.

3. Complexities of the Murrinh-Patha verb

Only very limited language data is available for Murrinh-Patha. In fact, the corpus of Bible translations used in the evaluation (see section 5) with approximately 22 000 words is the only easily available collection of texts so far. This is not enough for a statistical approach for a morphological analyzer. Moreover, this section presents some complexities of the Murrinh-Patha verb which show that a statistical approach, involving counts of full word forms or of morphemes, would face many difficulties even if there were enough text available.

Nouns in Murrinh-Patha may be case-marked and inflected with discourse markers, but are not difficult for morphological analysis. In contrast, verbs in Murrinh-Patha are morphologically very complex. They may consist of up to 9 morphemes and complex interdependencies between the morphemes exist. Additionally, phonological rules may apply when the morphemes combine. In the following, some selected examples of the complexities of the Murrinh-Patha verb are presented. For a more detailed overview, see Blythe (2009), Nordlinger (2010), Seiss (2011) and Street (1987).

Figure 1 shows the Murrinh-Patha verbal template. The lexical meaning of the verb is determined jointly by two morphemes, the so-called classifier stem (in slot 1) and lexical stem (in slot 5). The classifier stem encodes rather generic meaning and is traditionally glossed with a number. In contrast, the lexical stem encodes more specific meaning. Restrictions on the combinatorial possibilities exist, i.e. not every classifier stem can combine with every lexical stem and vice versa.

While lexical stems are noninflecting, classifier stems inflect for tense as well as for subject number and person. The inflection on the classifier stem is encoded in portmanteau forms. This results in more than 50 forms per paradigm, i.e. in forms which have a very different surface realization although underlyingly, the same classifier stem is used. An example involving classifier stem 13 in combination with the lexical stem *ngkardu* 'see' is given in (1).

(1) a. **bam** - ngkardu
1sgS.**13**.nFut - see
'I saw him/her.'

b. **ngube** - ngkardu - dha
1plS.**13**.PImpf - see - PImpf
'We saw him/her.'

Because of the different surface forms of the classifier stems, statistical methods which operate on the surface form of strings would require a very large amount of data. Other verbal template slots contribute to variation in the surface form of the verb, too. Body parts may be incorporated in slot 4 and adverbials and particles may be attached in slot 7 and 9. This means that even more different surface forms of a classifier and lexical stem combination exist which, without a detailed morphological analysis, cannot be linked to the same underlying information. The complexity of the verbal template thus results in a low frequency of surface form realizations in any given text.

A further complexity is added by a high degree of syncretism. As (1) and (2) show, the surface form *bam* can be classifier stem 13 or 18. However, only the classifier 18 can combine with the lexical stem *bat* 'fall'. Thus, in this case, *bam* can only be interpreted as classifier 18.

(2) **bam** - bat
1sgS.**18**.nFut - fall / *1sgS.**13**.nFut - fall
'I fell.'

This shows that detailed modeling of the morphology which takes the dependencies between the classifier stem and the lexical stem into account may thus help to restrict choices and so restrict overgenerating of the analyzer.

As has been stated above, the classifier stems are inflected for tense. Additionally, all tenses, except the non-future tense, require an additional tense marker in slot 6 which agrees with the tense marking on the classifier stem. For example, the verb in (3a) needs a future marker in slot 6 because the classifier stem is in future tense. No tense marker, or an alternative tense marker, e.g., a past imperfective marker as in (3b), is ungrammatical.

(3) a. ba - ngkardu - nu
1sgS.13.Fut - see - Fut
'I will see him/her.'

b. *ba - ngkardu - dha
1sgS.13.Fut - see - PImpf

While the tense marker dependencies are quite simple agreement dependencies, more complex dependencies exist as well in which the relative position of the morphemes to each other plays a role. This has lead to the claim that the Murrinh-Patha verbal morphology is a templatic system (Nordlinger, 2010). As can be seen in (4a), for example, the dual subject number marker *ngintha* normally attaches in slot 2, between the classifier and lexical stem. However, if an object marker is present as in (4b), the subject number marker has to be realized in slot 8, after the lexical stem. As (4c) shows, this is impossible if the object marker is not expressed in slot 2.

1	2	3	4	5	6	7	8	9
Class.	SubjN/ Obj	RR	IBP APPL	Lex	TNS	Adv/Prt	SubjN/ ObjN	Adv/Prt

Class: classifier stem, marked for tense, aspect & subject number
SubjN: subject number markers for dual & paucal subject
Obj/ObjN: object markers / object number markers
RR: reflexive / reciprocal marker
IBP: incorporated body part
Lex: lexical stem
TNS: tense marker
Adv/Prt: Adverbial / Particle

Figure 1: Murrinh-Patha verbal template (adapted from Blythe (2009))

(4) a. bam - ngintha - ngkardu
3sgS.13.nFut - du.f - see
'They two (non-siblings) saw him/her.'

b. bam - ngi - ngkardu - ngintha
3sgS.13.nFut - 1sgDO - see - du.f
'They two (non-siblings) saw me.'

c. *bam - ngkardu - ngintha
3sgS.13.nFut - see - du.f

Dealing with Murrinh-Patha data is further complicated by phonological processes which apply when morphemes are combined. These phonological processes thus disguise the fact that different surface realizations may involve the same underlying morpheme. For example, in (5), the same lexical stem *ngkardu* 'see' is used. However, the nasalization is lost in (5a) because of the preceding /m/ of the classifier stem. In contrast, the nasalization is retained in (5b) which involves the same classifier stem but which has an intervening subject number marker between classifier and lexical stem. As in this case the nasal /ng/ is not preceded by /m/, the nasalization is retained.

(5) a. *bam**k**ardu*
bam - ngkardu
3sgS.13.nFut - see
'He/she saw him/her.'

b. *bamngintha**ngk**ardu*
bam - ngintha - ngkardu
3sgS.13.nFut - du.f - see
'They two (non-siblings) saw him/her.'

The nasalization /ng/ is always lost after a nasal /m/ or /n/. However, even more complex rules exist in which the application of the rule depends on the lexical stem used. This is illustrated in (6) in which /my/ is changed to /nth/ for the lexical stem *yerr* 'look' but not for the lexical stem *yel* 'rain.'

(6) a. *mintherr*
mim - yerr
1sS.12.nFut - look
'I looked out/around.'

b. *kanamyel*
kanam - yel
3sS.4.nFut - rain
'It is raining.'

Phonological processes such as these are very common in Murrinh-Patha. The phonological processes together with the complexity of the verbal template make a statistical analyses of Murrinh-Patha difficult as many different surface forms may belong to one underlying representation. This is problematic for corpus studies as well as for statistical analysis used to build applications, such as morphological analyzers, parsers, machine translation etc. The following section thus proposes to model the complexities with a rule-based morphological analyzer for Murrinh-Patha which can handle all of these challenges.

4. Implementation of the Morphological Analyzer

The morphological analyzer has been implemented as a finite state analyzer using XFST and LEXC (Beesley and Karttunen, 2003). This system has been used because it offers a wide range of inbuilt mechanisms to handle complex cases such as those found in the Murrinh-Patha verb. This section discusses how these inbuilt mechanisms facilitate the modeling of the complexities on some selected examples. Section 4 then describes how this implementation is used to construct a robust morphological analyzer with a refined lookup strategy.

The concatenation of morphemes, both for the Murrinh-Patha verbs and the other parts of speech, is implemented with LEXC (Beesley and Karttunen, 2003) as two-level networks. It uses continuation classes for this purpose. An example is provided in (7), in which an entry from the lexicon ROOT is concatenated with an entry from the lexicon OBJECT and this with an entry from the lexicon LEX. The colon separates the two levels and "0" marks the zero morpheme.

In this way, the network in (7) can produce the output in (8). The surface string *bamngkardu* is associated with the information on the lower side, i.e. that this word is made up of a classifier stem *bam* which is classifier stem 13 inflected for third person singular non-future tense, a zero direct object marker and a lexical stem *ngkardu*.

(7) Lexicon ROOT
bam+class13+3P+sg+nFut:bam OBJECT;

Lexicon OBJECT
+3sgDO:0 LEX;

Lexicon LEX
+ngkardu+LS:ngkardu #;

(8) bamngkardu :
bam+class13+3P+sg+nFut+3sgDO+ngkardu+LS

The two-level approach makes it possible to distinguish between surface form and the associated information. This is especially helpful for Murrinh-Patha as the same classifier stem may have very different surface forms, depending on its inflection. Thus, while e.g., a stemmer would not be helpful for handling Murrinh-Patha verbs, this more advanced system of two-level morphology facilitates the use of statistical analyses of texts independently of the surface form.

As was discussed in the previous section, dependencies between morphemes in the Murrinh-Patha verbal template exist. These dependencies often concern non-adjacent template slots, i.e., they are long-distance dependencies. XFST offers a possibility to model these long-distance dependencies with the help of flag diacritics. Flag diacritics are special entities in XFST which add a kind of "short term memory" to keep track of what choices have been made before. Thus, as Beesley and Karttunen (2003, 341) explain, normally, "the transition from one state to the next depends only on the current state and the next input symbol". Using flag diacritics allows one to keep track of choices made earlier, so that certain transitions can also be constrained by choices made earlier.

(9) is a first simple example of the use of P- and R-type flag diacritics. The P-type flag diacritics are used to set a value, e.g., for tense, to positive. In contrast, the R-type flag diacritics require a value to have been set to positive to allow the respective combination.

In (9), the combination of flag diacritics can be used to model the long distance dependencies between the classifier form and the tense markers in slot 6. The lexicon Root lists the classifier stems. From this lexicon, the various classifier stem forms are sent to different lexicons to pick up their respective tense information. For example, *bam* carries non-Future information and is consequently sent to the lexicon NFUT to pick up the flag diacritic "@P.Tense.nFut@". This flag diacritic sets the value for the attribute 'Tense' to 'nFut', i.e., it remembers that *bam* is non-future tense. In contrast, *ba* receives the flag diacritic "@P.Tense.Fut@" to remember that its tense value is future.

When the corresponding tense markers are attached in the lexicon TENSE_SLOT6, the choices for the combination are constrained by the R-type flag diacritics which require a certain value for tense. The morpheme *-nu* can only attach to a future classifier stem form, i.e. this choice is marked with the flag diacritic "@R.Tense.Fut@" and consequently, only strings are possible which include the flag diacritic "@P.Tense.Fut@". Similarly, the first line in the lexicon TENSE_SLOT6 is marked by the flag diacritic "@R.Tense.nFut@" which specifies that this choice, i.e. no morpheme attaching in this slot, is only possible if the value of the feature "Tense" has been set to "nFut" before.

(9) Lexicon ROOT
bam NFUT;
ba FUT;

Lexicon NFUT
@P.Tense.nFut@ LEX;

Lexicon FUT
@P.Tense.Fut@ LEX;
...
...
Lexicon TENSE_SLOT6
@R.Tense.nFut@ #;
@R.Tense.Fut@:**nu**@R.Tense.Fut@ #;

In a similar way, the dependencies between the classifier and lexical stems can be modeled. The dependencies between the subject number and the object markers in slot 2 and 8 are more complex and thus involve an interplay of different flag diacritics. While this is more difficult, it can nevertheless be modeled reliably with flag diacritics.

The advantage of modeling these interdependencies as closely as possible is that this approach ensures that the resulting network just represents the valid combinations of verbal morphemes and does not overgenerate. This is especially important because of the large degree of syncretism which would, without a detailed modeling of the dependencies, lead to many different analyses for one item.

A further complexity of the Murrinh-Patha verb lies in the phonological rules that apply when morphemes are combined. This means that the sole concatenation of strings in fact often does not represent the surface form of the verb. XFST offers the possibility to formulate phonological rules which rewrite the concatenation of morphemes to the actual surface form of the verb. For example, (10) accounts for the data in (5) in which /ng/ is lost if it follows an /m/ or /n/. Thus, the actual network contains the surface form and associated information as specified in (12).

(10) [n g k –> k || m _ , n _]

(11) bamkardu :
bam+class13+3P+sg+nFut+3sgDO+ngkardu+LS

For more complex cases in which the application of the rule depends on the lexical stem as in (6) above, the lexical stem can be marked when it is concatenated with the other morphemes and the regular expression can take this marking into account. Thus, the output of the concatenation of morphemes for the data in (6) is as illustrated in (12). In (12a), the lexical stem *yel* does not trigger any phonological change. In contrast, the lexical stem *yerr* in (12b) triggers a change and is therefore marked especially with a capital letter /Y/. The application of the phonological rules in (13) then causes the /Y/ to be changed to /nth/ after /m/ and to /y/ in all other cases and so ensures the right surface form.

(12) a. kanam**y**el :
kanam+class4+nFut+3P+sg+nFut+yel+LS

Lexical items	Number of entries
all forms of 38 classifier stems	
incorporated body parts	60
incorporated adverbs / particles	15
lexical stems	898
lexical plus classifier stem combinations	1732
nouns	1140
borrowed nouns	101
adjectives	175
adverbs	41
interjections	55

Table 1: Number of lexical entries for the Murrinh-Patha morphological analyzer. Top: different morphemes of the verb; Bottom: stems for other parts of speech.

b. mim**Y**err :
mim+class12+nFut+1P+sg+nFut+yerr+LS

(13) [m Y – > n t h] .o. [Y – > y]

These inbuilt mechanisms provided by LEXC and XFST thus allow a reliable model of the details of the complexities of the Murrinh-Patha verb. The following section describes how these mechanisms can be put to use in a robust morphological analyzer with high precision and coverage.

5. Stepwise Lookup Strategy

The goal for the morphological analyzer is, on the one hand, to have high coverage, while on the other hand to only provide the right analyses, i.e., to have high precision. In order to achieve this goal, the morphological analyzer uses a refined lookup strategy in which constraints on the dependencies of the verbal morphemes are relaxed stepwise. For this, known combinations of morphemes are modeled with strict constraints. However, to also capture unknown combinations of morphemes, especially unknown combinations of classifier and lexical stem combinations and new lexical stems, the constraints are relaxed, so that these unknown combinations can also receive an analysis. In this way, new lexical item candidates can be detected and given a first analysis, which then have to be verified later in field work.

All lookup steps for the morphological analysis use the same underlying lexicon. The lexicon contains all forms of the 38 classifier stems as well as all forms of the functional morphemes, i.e. number markers, tense markers etc. These are rather restricted and well-described classes of morphemes. For the larger classes of morphemes, i.e. lexical stems, incorporated body parts, adverbials and particles, automatically extracted dictionary entries from Joe Blythe's toolbox[1] dictionary have been used. This was also the case for the other parts of speech, such as nouns, adjectives, adverbials and interjections. An overview over most of the lexicon entries is provided in Table 1.

[1]http://www.sil.org/computing/toolbox/

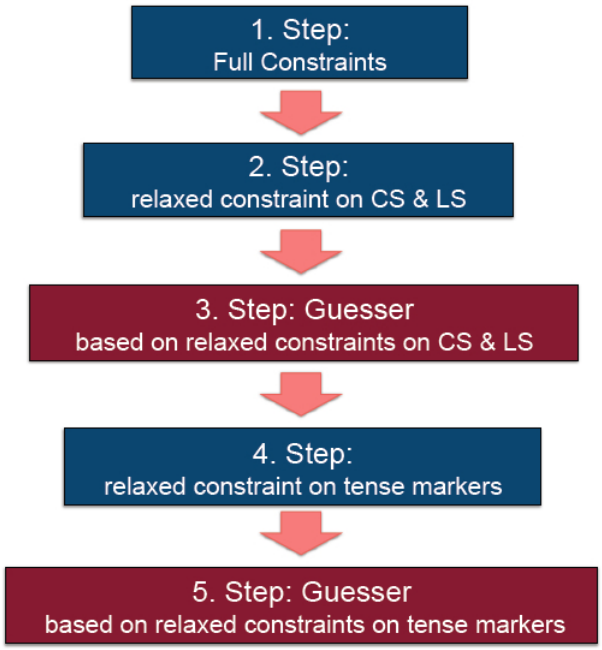

Figure 2: Refined lookup strategy for the morphological analyzer.

With this underlying lexicon, 5 different steps for a refined lookup strategy have been implemented in order to ensure a robust morphological analyzer. Figure 2 provides an overview over the steps of the lookup strategy. In a first step, all constraints governing morpheme combinations are implemented in the morphological analyzer. For the dependencies between lexical and classifier stems, for example, this means that only these verbs can be analyzed which were listed in the lexicon as possible lexical and classifier stem combinations. As Table 1 shows, these were 1732 different combinations.

As it is very likely that other combinations of lexical and classifier stems may also exist and that these have not been listed in the lexicon so far, the second step of the lookup strategy relaxes the constraint on the lexical and classifier stem combination. Technically, this means that the same finite state network is used, just without flag diacritics modeling the dependency between the classifier and lexical stem.

As lexical stems are the largest class of morphemes and not all lexical stems are listed in the lexicon, a morphological guesser for lexical stems is incorporated into the system as step 3. The morphological guesser thus only applies if the first two networks do not produce a solution.

The guesser for Murrinh-Patha lexical stems has been implemented as described by Beesley and Karttunen (2003) for guessers in general. The LEXC lexicon for lexical stems includes a placeholder entry which then in a network is replaced by all phonologically possible lexical stems. In Murrinh-Patha, lexical stems need to have at least one syllable and they usually start with a consonant. However, some lexical stems starting with the vowel 'a' are also attested.

These constraints have been integrated into the morphological guesser.

The guessed lexical stems may also undergo phonological changes. This means that different guesses may be possible for one input string. For example, if the lexical stem *ngkardu* 'see' were unknown, a possible guess for the input string *bamkardu* would be both, *kardu* and *ngkardu* as lexical stems in combination with the classifier stem 13.

Because of the various guesses due to the phonological rules, the morphological analyzer has to have step 2 and step 3 of the lookup strategy as separate steps. The guesser could also cover the known lexical stems in unknown combinations, but due to the phonological rules, the guesser produces various possible analyses. Separating the two steps by first relaxing the constraints on the classifier and lexical stem combinations and then guessing unknown lexical stems gives priority to already existing lexical stems. It so constrains the amount of possible analyses for the input strings with a known lexical stem.

For the steps 4 and 5 of the lookup strategy, the constraint on tense markers has been relaxed. Recall that it is claimed that a separate tense marker in slot 6 is mandatory for all tenses but the non-Future tense. However, inspection of the corpus texts and other language material revealed that this constraint is probably not as strict as described theoretically, i.e., the tense marker is sometimes not used with the other tenses as well. To cover these non-standard cases, the constraint on the tense markers has been relaxed for the lookup step 4. Additionally, a guesser without this constraint on the tense marker is used in step 5. Thus, in these steps new combinations of classifier and lexical stems and new lexical stems which could not be found in the previous steps can be detected.

This system is constrained to detecting more Murrinh-Patha verbs stepwise. For nouns, no guesser has been implemented as nouns in Murrinh-Patha do not have to be case-marked or carry other inflection. For this reason, a guesser for nouns is not helpful as all input strings could be potential nouns. As a result of the refined lookup strategy, however, most unanalyzable input strings will be nouns or other uninflecting parts of speech.

To sum up, the described lookup strategy makes the morphological analyzer stable and reliable. Relaxing the constraints stepwise ensures that the system does not overgenerate and at the same time has broad coverage. Thus, if the system is used to analyze a corpus, the system can detect new combinations of classifier plus lexical stem combinations and new lexical stems. In the following section, the system is evaluated against a small corpus of Bible translations.

6. Evaluation

Morphological analyzers can be evaluated by negative and positive testing. Negative testing indicates whether a morphological decomposition can be found for a given string of words and so measures pure coverage. Positive testing, on the other hand, indicates that the analyses which were found were correct, i.e., whether all possible analyses were found and whether the analyzer does not overgenerate, i.e., it measures precision and recall.

	all	**1.**	**2.**	**3.**	**4.**	**5.**
Types – dev. corpus	97.3	65.4	5.0	22.9	1.6	2.4
Text – dev. corpus	98.9	88.7	1.0	5.3	1.9	2.0
Types – test corpus	96.0	65.0	4.4	22.6	1.4	2.6
Text – test corpus	98.0	86.3	1.0	6.9	1.6	2.3

Table 2: Negative test results of the development and test corpus; *text* refers to running text, *types* to lists of words extracted from the respective corpora; the column *all* provides the overall coverage of the morphological analyzer; columns 1. to 5. refer to the coverage of the respective step in the lookup strategy.

For the evaluation of the morphological analyzer, a small corpus of Murrinh-Patha Bible translations (Wycliffe Bible Translators, 1990) has been used. The text itself consists of 22 348 words. For the evaluation, the text has been divided into 2 corpora, a development corpus (approx. 80 %) and a test corpus.

The development corpus consists of 17 950 words. From these, a word list has been generated, which amounts to 2388 types. For this development corpus, error analyses have been carried out and as many new lexical entries as possible have been added. For example, 101 borrowed English nouns and names which figure prominently in Bible texts needed to be added.

The details for negative testing of the development corpus, depending on the various lookup strategies described in the previous section, are presented in Table 2. For the development corpus, the analyzer has an overall coverage of 98.9% for running text, and 97.3% for types.

A close inspection of the unanalyzable input strings shows that most seem to be other parts of speech such as nouns or adjectives, but their meaning and function could not be established without doubt even by manual inspection and have thus not been added to the morphological analyzer. However, they can be tested in future field work.

A small portion of the unanalyzable items are also due to errors in the corpus, mainly because of erroneous script recognition. These will be fixed in the corpus.

When looking at the statistics for the negative testing in Table 2 in more detail, a comparison of the results for the development corpus between types and running text shows that the morphological analyzer already has a satisfactory coverage of frequent items as step 1, which models all constraints, covers 88.7% of all running text. However, step 1 only accounts for 65.4% for types. This discrepancy shows that there are in fact items which are not covered by step 1 but that these tend to be low-frequency items.

Despite this discrepancy of the coverage of step 1 between running text and types, the overall coverage for running text and types does not differ greatly. This is mainly the result of step 2 and step 3 of the lookup strategy. Step 2 (the morphological analyzer with relaxed constraints on the combination of classifier and lexical stems) can account for 5%

of the types, while it only accounts for 1% of the running text. This means that the unknown classifier and lexical stem combinations are also the low frequent ones. Similarly, step 3 (the morphological guesser) finds morphological analyses for 22.9% of all types compared to 5.3% of the analyses of the running text. Here again, the unknown lexical stems are the low frequency ones.
Finally, steps 4 and 5 of the lookup strategy account for approximately 4% of the coverage for each, running text and types. This shows that separate tense markers are usually used as described theoretically, but that in some cases they are also dropped.
For the test corpus, a second part of the Bible translations has been used. The test corpus comprises 4398 words and 919 types. The results are also provided in Table 2. Compared to the development corpus, the coverage is smaller, but not considerably. Especially the coverage of the first strategy of the running text is very high, which again shows that the frequently used items are covered by the morphological analyzer.
The stepwise strategy makes an interpretation of the output of the analyzer very easy. If a morphological analysis is the output of step 2 or step 4, it involves unknown classifier and lexical stem combinations. From this corpus (test and development corpus together), 113 new candidates for classifier and lexical stem combinations have been found.
The output of the morphological guessers, i.e., steps 3 and 5 which are built to extract new lexical stem candidates, is more difficult to interpret. First, the guessers still propose many different guesses per input string. This is due to the phonological rules which may apply, but also due to material in the input string which could be part of the lexical stem or separate morphemes. (14) shows some selected candidates for the input string *banninthalangyu*.

(14) banninthalangyu :
ban +class17+3P+sg+nFut ...
...+ninthalangyu+Guess+LS
...+ninthalang+Guess+LS+part
...+winthalang+Guess+LS+part
...+du.m.Nsibl.S+lang+Guess+LS+part

The input string *banninthalangyu* can be decomposed into the classifier stem *ban*, but then various possibilities exist for further decomposition. *ninthalangyu* could be the lexical stem as a whole, or it could be further decomposed into the lexical stem *ninthalang* and the particle *yu*. As a third possibility, a phonological rule could apply because /w/ changes to /n/ before /n/ and consequently, the lexical stem could be *winthalang*. Most likely, however, the *nintha* is a dual subject marker, so that the lexical stem is *lang*.
When manually investigating the list of candidates, the most probable guess is usually obvious. However, having an algorithm which extracts the most probable guess from the number of candidates automatically would make the manual work easier. As a first heuristic, the analysis which decomposes the input string into the highest number of morphemes, i.e., shortest match, can be used to extract the guessed lexical stem. However, the guesses for one input string may also depend on the guesses of the other input strings, as it would be preferable that new lexical stems are attested in more than one input string. Therefore, an algorithm should take both into account, the number of morphemes of the input string and the co-occurrence of the guessed lexical stem in other input strings. This automatic algorithm is left for future research.
A second problem of the morphological guesser is that not all input strings for which a guess can be found may be verbs. A brief inspection of the data indicates that most guesses seem to be reasonable, but that also some may be analyzed wrongly.
This points to the more general problem of positive testing, which, in contrast to negative testing, cannot be carried out easily. Positive testing requires a Gold Standard, but the resources for this are not available right now, as creating a Gold Standard requires extensive field work. Secondly, it is very difficult to comprise a Gold standard for the Murrinh-Patha morphological analyzer as due to syncretism many different analyses for one input string are often possible.

7. Conclusion

This paper presented the challenges of a morphological analysis of Murrinh-Patha verbs and proposed a rule-based analyzer which was implemented using XFST and LEXC. The complexities of the Murrinh-Patha verb require a morphological preprocessing for almost all NLP applications. The output of the morphological analyzer is a simple binary file. As the system has not been developed with a specific purpose in mind, it is resource independent and can thus be integrated into many different applications.
Currently, the morphological analyzer is used in an XLE parser (Crouch et al., 2011) for Murrinh-Patha. The XLE parser is integrated into the ParGram community with standardized morpho-syntactic features and analyses (Butt et al., 1999).
The morphological analyzer together with the parser are used in an electronic dictionary and in a simple translation system from English to Murrinh-Patha (Seiss and Nordlinger, 2011). These systems are primarily intended as learning resources for non-Murrinh-Patha speakers. A high degree of accuracy is therefore needed, which can only be accomplished with a rule-based system.
The refined lookup strategy can also be very helpful in these applications. For example, the electronic dictionary performs a morphological analysis to decompose the input string into classifier and lexical stem combination and then presents the user with the meaning for this combination. In this context, the lookup strategy can be used for more informative feedback. If an input string cannot be decomposed into a classifier and lexical stem combination which has an entry in the dictionary, the electronic dictionary can at least offer some more hints, e.g., by giving the decomposition with an unknown classifier and lexical stem combination or by providing a guess for the lexical stem.
In sum, the paper showed how such complex systems as the Murrinh-Patha verbal system can be modeled with rule-based methods reliably and how a robust morphological analyzer can be implemented based on these methods. This implementation can now be put to use in many different applications.

8. Acknowledgements

I would like to thank Rachel Nordlinger and Joe Blythe who provided me with data and information about the language. Many thanks also go to my supervisor Miriam Butt.

9. References

Iñaki Alegria, Xabier Artola, Kepa Sarasola, and Miriam Urkia. 1996. Automatic morphological analysis of Basque. *Literary and Linguistic Computing*, 11(4):193–203.

Mohammed Attia, Pavel Pecina, Antonio Toral, Lamia Tounsi, and Josef van Genabith. 2011. An open-source finite state morphological transducer for modern standard Arabic. In *Proceedings of the 9th International Workshop on Finite State Methods and Natural Language Processing*, pages 125–133, Blois, France. Association for Computational Linguistics.

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI Publications, Stanford.

Kenneth R. Beesley. 1996. Arabic finite-state morphological analysis and generation. In *Proceedings of the 16th conference on Computational linguistics, COLING 96*, pages 89–94.

Joe Blythe. 2009. *Doing Referring in Murriny Patha conversation*. Ph.D. thesis, University of Sydney.

Tina Bögel, Miriam Butt, Annette Hautli, and Sebastian Sulger. 2007. Developing a finite-state morphological analyzer for Urdu and Hindi. In *Proceedings of the Sixth International Workshop on Finite-State Methods and Natural Language Processing, Potsdam*.

Miriam Butt, Tracy Holloway King, María-Eugenia Niño, and Frédérique Segond. 1999. *A Grammar Writer's Cookbook*. CSLI, Stanford.

Dick Crouch, Mary Dalrymple, Ron Kaplan, Tracy King, John Maxwell, and Paula Newman. 2011. XLE documentation. URL: http://www2.parc.com/isl/groups/nltt/xle/doc/xle_toc.html.

Mans Hulden. 2009. Foma: A finite-state compiler and library. In *Proceedings of the 12th Conference of the EACL Demonstration Session*, pages 29–32.

Kimmo Koskenniemi. 1983. Two-level morphology: A general computational model for word-form recognition and production. Publication 11, University of Helsinki, Department of General Linguistics, Helsinki.

Septina Dian Larasati, Vladislav Kuboň, and Daniel Zeman. 2011. Indonesian morphology tool (MorphInd): Towards an Indonesian corpus. In Cerstin Mahlow and Michael Piotrowski, editors, *Systems and Frameworks for Computational Morphology: Proceedings of the Second International Workshop, SFCM 2011, Zürich*, pages 119–130.

Krister Lindén and Tommi Pirinen. 2009. Weighted finite-state morphological analysis of Finnish compounding with HFST-LEXC. In *Proceedings of the 17th Nordic Conference of Computational Linguistics NODALIDA 2009*, volume 4, pages 89–95.

Krister Lindén, Erik Axelson, Sam Hardwick, Tommi Pirinen, and Miika Silfverberg. 2011. HFST—framework for compiling and applying morphologies. In Cerstin Mahlow and Michael Piotrowski, editors, *Systems and Frameworks for Computational Morphology: Proceedings of the Second International Workshop, SFCM 2011, Zürich*, pages 67–86.

Karine Megerdoomian. 2004. Finite-state morphological analysis of Persian. In *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages. Coling 2004, University of Geneva*.

Rachel Nordlinger. 2010. Verbal morphology in Murrinh-Patha: Evidence for templates. *Morphology*, 20(2):321–341.

Melanie Seiss and Rachel Nordlinger. 2011. An electronic dictionary and translation system for Murrinh-Patha. In *Proceedings of the EUROCALL 2011 conference, University of Nottingham*. to appear.

Melanie Seiss. 2011. Implementing the morphology-syntax interface: Challenges from Murrinh-Patha verbs. In Miriam Butt and Tracy H. King, editors, *Proceedings of the LFG2011 Conference*, pages 482–502, Stanford. CSLI Publications.

Chester Street. 1987. *An Introduction to the Language and Culture of the Murrinh-Patha*. Pacific Linguistics, Port Keats.

Heli Uibo. 2005. Finite-state morphology of Estonian: Two-levelness extended. In *Proceedings of RANLP 2005, Borovets*, pages 580–584.

Ramasamy Veerappan, P J Antony, S Saravanan, and K P Soman. 2011. A rule based Kannada morphological analyzer and generator using finite state transducer. *International Journal of Computer Applications*, 27(10):45–52.

Wycliffe Bible Translators. 1990. Murrinh ngarra Jesus nukunu: The story about Jesus, book 1.

Shlomo Yona and Shuly Wintner. 2008. A finite-state morphological grammar of Hebrew. *Natural Language Engineering*, 14(2):173–190.