Proceedings of the Australasian Language Technology Workshop, Vol 6 (2008), pages 142-150

A Two-Level Morphological Analyser for the Indonesian Language

Femphy Pisceldo, Rahmad Mahendra & Ruli Manurung
Faculty of Computer Science
University of Indonesia
fep40@ui.edu,rama42@ui.edu,maruli@cs.ui.ac.id

I Wayan Arka
Department of Linguistics
Australian National University
wayan.arka@anu.edu.au

Abstract

This paper presents our efforts at developing an Indonesian morphological analyser that provides a detailed analysis of the rich affixation process[1]. We model Indonesian morphology using a two-level morphology approach, decomposing the process into a set of morphotactic and morphophonemic rules. These rules are modelled as a network of finite state transducers and implemented using **xfst** and **lexc**. Our approach is able to handle reduplication, a non-concatenative morphological process.

1 Introduction

Morphology is the study of the way that words are built up from smaller units called morphemes, the minimal meaning-bearing units in a language (Jurafsky, 2000). For example, the English word *kind* consists of a single morpheme (the root word *kind*) whilst the word *players* consists of three morphemes: *play*, *-er* and *-s*. The morphemes *kind* and *play* can stand alone as words, while affixes *-er* and *-s* must appear bound to another morpheme.

By applying a set of morphological rules, we can produce not just morphemes but also other information relating to the words; for example, the grammatical category of the whole word as well as the subcategorisation frame of the word if it is a verb. This process is called morphological analysis. In this respect, the value of a morphological analyser would be twofold: from a theoretical (linguistic) viewpoint, it is a very useful tool for linguistic modelling and for testing certain analyses. On the other hand, from a practical viewpoint, it supports many applications, e.g. information retrieval, search engines, and machine translation, among others.

There is currently some interest in developing morphological tools for the Indonesian language. In previous work, Siregar (1995) and Adriani et al. (2007) discuss the development of Indonesian stemmers that recover a root from an affixed word, implemented procedurally. However, stemmers are of limited use as they do not provide any more linguistic information beyond the stem. Hartono (2002) presents an initial version of a morphological analyser developed with PC-KIMMO, but unfortunately, it does not handle reduplication, a key aspect of Indonesian morphology. The morphological analyser that we are developing and that we describe here is designed to be able to handle the rich semantic, lexical and grammatical information associated with words and word formation in NLP applications.

In Section 2, we first discuss Indonesian morphology, followed by a brief explanation of two-level morphology in Section 3. Sections 4 and 5 present our work in applying two-level morphology for the Indonesian language. Finally, Section 6 presents the results of some evaluations we carried out on our developed analyser.

2 Indonesian Language Morphology

Indonesian is a variety of Malay, which belongs to the Austronesian language family (Gordon, 2005; Sneddon, 2003). It is spoken by about 190 million people in Indonesia and other parts of the world[2].

[1] This research is part of a collaborative research project funded by ARC Discovery Grant DP0877595.

[2] According to *Biro Pusat Statistik*, as of 2004.

Words in Indonesian are built from their roots by means of a variety of morphological operations including compounding, affixation, and reduplication. Indonesian concatenative morphology regulates how a stem and its affixes glue together, while a non-concatenative one combines morphemes in more complex ways.

Affixes in Indonesian can be classified as four categories (Alwi et al., 2003). Prefixes precede the base form, i.e. *meN-*, *di-*, *peN-*, *per-*, *ke-*, and *ter-*. Suffixes follow the base form, i.e. *-kan*, *-an*, and *-i*. Infixes are inside the base form, i.e. *-el-*, *-em-*, and *-er-*. Circumfixes wrap around the base form. While circumfixes formally are combinations of allowed prefixes and suffixes, they have to be treated as discontinuous units for semantic and grammatical reasons. In our Indonesian morphological analyser, we have handled prefixes, suffixes, and circumfixes.

Indonesian non-concatenative morphology refers to reduplicated morpheme forms. Reduplicated words based on morpheme regularity are grouped into full reduplication (e.g., the word *buku-buku* is derived from the stem *buku*) and partial reduplication of different kinds. The latter includes reduplicated stems with affixes (e.g. the word *buah-buahan* is derived from stem *buah*, *bertingkat-tingkat* is derived from stem *bertingkat*) and various (often rather irregular) reduplications (e.g. *sayur-mayur* is derived from *sayur*).

Indonesian affixation and reduplication are illustrated in the following example. From the stem *pukul*, we can derive words like *pemukul* (by concatenating the prefix *peN-*), *memukuli* (by concatenating the circumfix *meN-i*), and *pukulan-pukulan* (by first concatenating the suffix *–an*, then applying full reduplication). Other examples include *dipukul, pemukulan*, and *berpukul-pukulan*.

All kinds of regular word-formation with or without reduplication are recognized in this research.

Morphology generally makes a distinction between inflectional and derivational processes. In the previous example, the formation of *memukuli* appears to be 'inflectional' as the formation does not change the category of the stem, and the formation of *pemukul* and *pukulan-pukulan* is derivational because the derived words are nouns while the stem *pukul* is a verb. However, the distinction between inflection and derivation, particularly in Indonesian, is not always clear cut. The formation of verbs such as *memukuli* from *pukul* is arguably derivational in nature because the new words have quite different lexical properties, even though both the new verbs and the stems are of the same category (i.e. 'verb').

3 Two-Level Morphology

A morphological analyser can be used to process a list of morphemes in a lexicon to yield fully derived words. In the other direction, it can be used to identify affixes and roots from derived words. For example, taking the word *membaca* as input, an Indonesian morphological analyser should produce the string *baca*`+Verb+AV`, indicating that the active verb *membaca* is derived from *baca*.

However, it is difficult to accomplish morphological analysis in one step only. In Indonesian, affixation does not consist of just fusing the affix with the stem. Modification of phonemes may be necessary, e.g. the concatenation of *meN+putar* will produce word *memutar*, while the result of *meN+siram* is *menyiram*.

To solve this problem, we adopt a two-level morphology model. Two-level morphology is based on finite-state transducers (Koskenniemi, 1983). In practice, this model has been successfully applied to several languages such as English, Finnish, Japanese, Russian, and French. It is not only useful to account for various concatenative morphologies, but is also able to handle non-concatenative processes, as has been developed in Malagasy tokenization and morphological analysis (Dalrymple et al., 2006).

4 Design

Our design for an Indonesian morphological analyser is divided into two components: **morphotactic** rules that explain which classes of morphemes can follow other classes of morphemes inside a word, and **morphophonemic** rules which model the changes that occur in a word. The rules in each component are typically applied in parallel. Additionally, these rules are combined with a lexicon of stems in order to complete the full design.

A word, in order to be analysed, will follow the path lexicon → morphotactic rules → morphophonemic rules→ surface. Before the result of the morphological analyser appears at the surface, it will follow the lexicon path to determine the actual morpheme of that word. After moving from the

lexicon, that word will be analysed by morphotactic and morphophonemic rules. Only after finishing the process in morphotactic and morphophonemic rules, will the result of morphological analyser for that word be delivered.

The explanation below will explore, in a little more depth, the design of lexicon, morphotactic rules, and morphophonemic rules.

4.1 Lexicon design

Our lexicon equates to a set of Indonesian stem words. Affixes are not stored in the lexicon as they are already accounted for by the morphotactic rules.

For our initial design, our lexicon is divided into four classes, i.e. verbs, nouns, adjectives, and 'etc', which includes all other stems, e.g. pronouns, adverbs, numbers, and particles. Clustering these word classes together is certainly a large oversimplification, and one which we hope to address in future revisions.

4.2 Tag design

The design of tags has become very important in the development of morphological analysers, since the tags will deliver linguistic information that occurs on a word being analysed.

In our research, the tags to be designed can be divided into normal tags and special tags. Normal tags can be output by the morphotactics component without any circumstances, whilst special tags only occur if the involved stems are associated with specific markers in the lexicon. The normal tags are +VERB, +NOUN, +ADJ, +BAREVERB, +BARENOUN, +BAREADJ, +BAREETC, +AV, +PASS, +UV, and +REDUP, whilst the special tags are +CAUS_KAN, +APPL_KAN, +CAUS_I, +APPL_I, +ACTOR, +INSTRUMENT.

Tags like +VERB, +NOUN, and +ADJ give the part of speech (POS) information for fully inflected words, whereas +BAREVERB, +BARENOUN, +BAREADJ, and +BAREETC give the POS information for stems.

Other tags provide important linguistic information, such as voice, a grammatical category playing a central role in Indonesian syntax, e.g. +AV, which indicates active voice, +PASS, which indicates passive voice, and +UV, which is reserved for undergoer voice.

Furthermore, +REDUP is a tag indicating reduplication. +CAUS_KAN, +CAUS_I, +APPL_KAN, and +APPL_I show whether words are causative or applicative with respect to their suffixes. The last two tags (+ACTOR and +INSTRUMENT) indicate either it is an actor or an instrument that is being brought with the construction of that word.

4.3 Morphotactic rules

In designing a morphological analyser, morphotactic rules are crucial to model how two or more morphemes can be merged.

Based on (Alwi et al., 2003), the morphotactic rules for Indonesian can be classified into 13 classes. Ten of these classes are determined based on which suffixes are merged with the stem, while the other three are reduplication cases. The first ten classes can be identified as concatenative morphology, while the other three are nonconcatenative morphology.

In our design, the ten classes that belong to concatenative morphology are the morphotactic rules for the prefixes *meN-*, *peN-*, *di-*, *per-*, *ber-*, *ter-*, and *ke-*, and the morphotactic rules for the suffixes *-an*, *-kan*, and *-i*. Some example cases that belong to these 10 classes are as follows:

- *meN* + stem(verb, noun, adjective or etc) + *kan* → Verb. Example: *membersihkan* (*meN+bersih+kan*). In this example, the word *bersih* is an adjective. After merging with *meN-kan*, the resulting word is a verb.
- *peN* + *ber* + stem(verb, noun, adjective or etc) + *an* → Noun. Example: *pembelajaran* (*peN+ber+ajar+an*). In this example, the word *ajar* is a verb. After merging with *peN-ber-an*, the resulting word is a noun.
- *ke* + *ber* + stem(verb, noun, adjective or etc) + *an* → Noun. Example: *keberhasilan* (*ke+ber+hasil+an*). In this example, the word *hasil* is a noun. After merging with *ke-ber-an*, the resulting word is also a noun.
- stem(verb, noun or adjective) + *i* → Verb. Example: *terangi* (*terang+i*). In this example, the word *terang* is an adjective. After merging with *–i*, the resulting word is a verb.

The last three classes, which belong to the category of nonconcatenative morphology, are the morphotactic rules for full reduplication, partial reduplication, and affixed reduplication. Some example cases that belong to these three classes are as follows:

- Full reduplication without affixation. Example: *buku-buku*. In this example, the word *buku-buku*, which is a noun, is generated from the word *buku*, which is also a noun.
- Full reduplication with *ke-an*. This case has the following rule: reduplication of (*ke*+stem (bare verb, adjective or etc)+*an*) → Noun. Example: *kekayaan-kekayaan* (reduplication of *ke+kaya+an*). In this example, the word *kaya* is an adjective. After the morphological process, the resulting word is a noun.
- Partial reduplication with *ber-*. This case has the following rule: *ber*+ reduplicated stem (bare verb) → Verb. Example: *berlari-lari* (*ber+lari-lari*). In this example, the word *lari* is a verb. After the morphological process, the resulting word is also a verb.
- Affixed reduplication with *meN-*. This case has the following rule: stem (bare verb) +*meN*+stem (bare verb) → Verb. Example: *tanam-menanam* (*tanam-meN+tanam*). In this example, the word *tanam* is a verb. After the process, the resulting word is also a verb.

During the stage of morphotactic rules, there are several steps that must be followed in order to complete the process. Those steps include the addition of prefixes and preprefixes, the addition of stems and part-of-speech, the addition of suffixes, and the final processing of additional tags. After finishing all of these steps, the process moves on to the morphophonemic process.

4.4 Morphophonemic rules

All the rules that define how two or more morphemes can merge have been designed in morphotactic rules. However, the merging process is still not completed; hence we still have to define what changes have to be made after these morphemes merge. For these issues, we define morphophonemic rules that define the phonetic changes that occur.

In Indonesian, these rules can generally be divided into two groups. The first group consists of four rules that model the phonetic changes in stems, whereas the second group consists of seven rules that model the phonetic changes in affixes.

Based on (Alwi et al., 2003), the four rules in the first group are:

- */k/* replacement with */ng/* if the word starts with *meN-* or *peN-*. Example: *meN+kantuk* → *mengantuk*.
- */s/* replacement with */ny/* if the word starts with *meN-* or *peN-*. Example: *peN+sebaran* → *penyebaran*.
- */p/* replacement with */m/* if the word starts with *meN-* or *peN-*. Example: *peN+pakai* → *pemakai*.
- */t/* replacement with */n/* if the word starts with *meN-* or *peN-*. Example: *meN+tertawakan* → *menertawakan*.

Note that the rules above only change the stem-initial segment. To complete the replacement processes, the following seven rules from the second group are needed to get the correct prefix forms:

- */N/* deletion if *meN-* is followed by */l/, /m/, /n/, /r/, /y/, /w/, /t/, /s/, /p/, /k/* or if there is *peN-* followed by */l/, /m/, /n/, /r/, /d/, /w/, /t/, /s/, /p/, /k/*. Example: *meN+lukis* → *melukis*.
- */r/* deletion if there is *ber-, ter-*, or *per-* followed by */r/* or the word that its first syllable ended with */er/*. Example: *ber+runding* → *berunding*.
- */N/* replacement with */n/* if there is *meN-* followed by */d/, /c/, /j/, /sy/* or there is *peN-* followed by */d/, /c/, /j/*. Example: *peN+jual* → *penjual*.
- */N/* replacement with */m/* if there is *meN-* or *peN-* followed by */b/, /f/*. Example: *peN+buru* → *pemburu*.
- */N/* replacement with */nge/* if there is *meN-* followed by a word that has only one syllable. Example: *meN+rem* → *mengerem*.
- */N/* replacement with */l/* if there is *peN-* followed by the word *ajar*. Example: *peN+ajar* → *pelajar*.
- */r/* replacement with */l/* if there is *ber-* followed by the word *ajar*. Example: *ber+ajar* → *belajar*.

After all sub-processes invoked by the rules in the first group and the second group are parallel-

ized, then the whole morphophonemic process is finished.

The design of morphophonemic rules for reduplication is very much the same as the one in affixation, since basically the morphophonemic process in reduplication occurs in the affixation part of reduplication.

However, several rules, in particular those that model morphophonemic processes where both affix and stem undergo changes, had to be revised to account for their behaviour when applied to reduplicated forms. For example, the morphophonemic rules '*/k/* replacement with */ng/*' and '*/N/* deletion', which work in tandem, were originally defined as '*/k/* deletion' and '*/N/* replacement with */ng/*'. However, this approach would not have worked for cases involving reduplication. For example, the word *mengotak-ngotakkan* will not be analysed properly if the morphological analyser uses the rules '*/k/* deletion' and '*/N/* replacement with */ng/*'. The word *mengotak-ngotakkan* is generated from the stem *kotak* that is modified with affixed reduplication *meN-kan*. If '*/k/* deletion' and '*/N/* replacement with */ng/*' are being used, the word will become *mengotak-otakkan* which is not valid. This is why, for the sake of reduplication, the rule is changed to '*/k/* replacement with */ng/*' that is parallelized with '*/N/* deletion'. Consequently, *mengotak-ngotakkan* can be properly generated.

5 Implementation

This Indonesian morphological analyser is implemented in **`xfst`** and **`lexc`** (Beesley & Karttunen, 2003). The morphotactic rules are implemented in **`xfst`** while morphophonemic rules are implemented in **`lexc`**.

5.1 Morphotactic rules implementation

The morphotactic rules can be illustrated as the finite-state automata shown in Figure 1. Valid Indonesian words, i.e. those that are constructed through legal morphological processes, are accepted by the automata, whereas invalid words are rejected.

Starting from the root, each state defines the next possible state whilst emitting (or consuming) a certain symbol. In **`lexc`**, these states are called continuation classes. All continuation classes reachable from the root represent prefixes and *pre-*prefixes. The distinction between the two is necessary to encode the possible morphological variations that take two prefixes, e.g. *memper-*, *diper-*. From there, the next continuation class is the **`Stem`**, where the root word is emitted or consumed. This is subsequently followed by several classes representing possible suffixes, but there are also **`Redup1`** and **`Redup2`** classes that appear before and after the suffixes. Their function is to handle reduplication (Section 5.3). Lastly, the **`TagEmit`** class processes

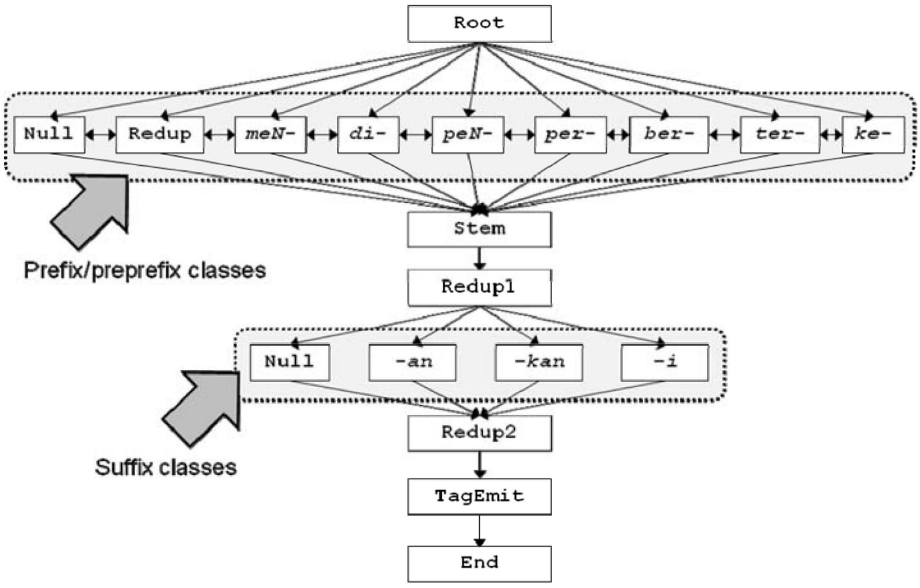

Figure 1. Illustration of Process Flow

all tags not yet handled by preceding classes (Section 4.2).

Throughout this morphotactic process, we extensively employ *flag diacritics*, a crucial feature of **lexc** which approximates the power of feature structures, i.e. being able to specify certain constraints to ensure that only valid paths of the network may be traversed. One benefit of this approach is the maintenance of a compact network representation. There are three flag diacritics used in our model: positive setting (**@P.feat.val@**), required test (**@R.feat.val@**), and disallow test (**@D.feat.val@**). Using these diacritics, we are able to stipulate values and constraints of certain aspects, which must be consistent throughout a path.

For example, the prefix *meN-* may combine with the suffix *–kan* and *–i*, but not *–an*. We can represent this with a feature, e.g. **PREF**, which is set to **meN** at the appropriate (pre)prefix state (**@P.PREF.meN@**). Suitable tests are then added to the suffix states, e.g. if the current path went through the *meN-* prefix state, **@R.PREF.meN@** stipulates that a suffix can be applied, whereas **@D.PREF.meN@** prevents a suffix from being applied.

To further exhibit the usage of flag diacritics in our morphological analyser, we provide an example of analysing the word *memukuli*, which consists of the prefix *meN-*, followed by the stem *pukul*, and the suffix *–i*. Figure 2 shows the path through the network used to analyse this word.

Starting from the root, the analyser parses the prefix *meN-* in *memukuli* and appropriately sets the flag diacritic for the feature **PREF**. This can be seen by the following **lexc** rule snippet:

```
LEXICON PrefixMeN
<[0 .x. m e "^N"] "@P.PREF.meN@"> Stems;
```

The **LEXICON PrefixMeN** line defines the state in the network that represents the meN- prefix. The next line defines a state transition where, by emitting or consuming the meN- prefix, the analyser can proceed to the **Stems** continuation class. Additionally, the **PREF** flag is set to **meN**. In reality, the **PrefixMeN** state has five other possible transitions, but they are not presented here.

Subsequently, at the **Stems** state, the analyser positively sets the flag diacritic for the feature **STEM** with value **verb**. Various required and disallow test flags will be called at the **Redup1**, **SuffixI**, **Redup2**, and **TagEmit** stages. However, for now, we focus on the **SuffixI** state, detailed in the following **lexc** rule **snippet**:

```
LEXICON SuffixI
<"+Verb":i "+AV":0 "@D.PREPREF@" "@D.REDUP@"
["@R.STEM.Vrb@"|"@R.STEM.Nom@"|"@R.STEM.Adj@"]
"@R.PREF.meN@" "@P.SUFF.i@"> Redup2;
```

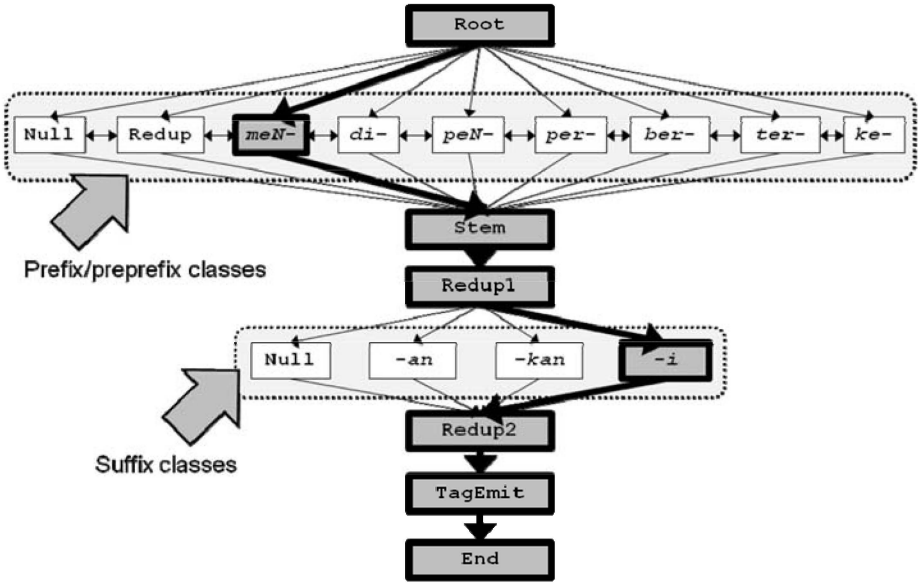

Figure 2. The path for *memukuli*

This rule represents a valid transition from the **SuffixI** state to the **Redup2** state by parsing the –i suffix and emitting the +**VERB** and +**AV** tags (**"+Verb":i "+AV":0**). Additionally, the diacritic tests **"@D.PREPREF@"** and **"@D.REDUP@"** stipulate that this transition can only be taken if the **PREPREF** and **REDUP** flags are not set, the disjunction **["@R.STEM.Vrb@"|"@R.STEM.Nom@"|"@R.STEM.Adj@"]** states that the **STEM** flag must be either **Vrb**, **Nom**, or **Adj**, and the **"@R.PREF.meN@"** test requires that the **PREF** flag is set to **meN**. Finally, **"@P.SUFF.i@"** states that the **SUFF** flag is set to **i**. Should any of these tests fail, the transition cannot be taken. If no other alternative paths are available to reach the **End** state, the word is rejected.

5.2 Reduplication Process

As discussed above in Sections 2 and 4, Indonesian morphology includes the non-concatenative process of reduplication. Handling this with pure regular grammars as implemented by finite state automata is very difficult. Thus, we employ the **compile-replace** feature in **xfst** (Beesley & Karttunen, 2000). This feature allows the repetition of arbitrarily complex sublanguages by specifying the brackets **"^["** and **"^]"** to mark the domain of reduplication. The right bracket is also augmented with **^2** to indicate duplication; thus, the full annotation is **"^["** and **"^2^]"**. Given this network definition, xfst compiles and post-processes these annotations to produce a new network where the appropriate reduplications have been carried out. For example, **"^[buku^2^]"** would eventually be compiled to become **bukubuku**.

Thus, the idea is to insert the **"^["** and **"^2^]"** annotations in the right places. Due to the various types of reduplication in Indonesian (see Section 2), the rules for reduplication can be found at the **Redup** (pre)prefix state and the **Redup1** and **Redup2** states. The **Redup** prefix state emits the opening **"^["** brackets and sets an appropriate flag as a reminder that the closing brackets are required. The **Redup1** state is responsible for closing partial and affixed reduplications, i.e. where the suffix is not included in the reduplication, whilst the **Redup2** state is responsible for closing full reduplication, i.e. where the suffix is part of the reduplication process. Both **Redup1** and **Redup2** states check for the value of the **REDUP** flag as set by the **Redup** prefix state.

5.3 Morphophonemic rules implementation

The full transducer composes the morphotactic and morphophonemic rules. As a result, the output of the morphotactic rules implementation serves as the input to the morphophonemic rules implementation.

From the example in Section 5.1 regarding the implementation of morphotactic rules, it can be observed that the string *me^Npukuli* has been produced. However, the morphological analysis process is still incomplete since *me^Npukuli* is not a valid word in Indonesian. In order to complete the process, the morphophonemic rules must be implemented. Since the full transducer composes the morphotactic and morphophonemic rules, the word *me^Npukuli* becomes the input for the morphophonemic process.

The implementation of morphophonemic rules differs slightly from the implementation of morphotactic rules. For morphotactic rules, there are several steps that can be illustrated as a flow of process. However, the implementation of morphophonemic rules generally implies the rules itself. Each rule is defined as a replacement rule that will collaborate with other rules through composition or parallelization.

All rules in Section 4.4 have been implemented. However, due to space constraints, we will only explain the implementation of rule '**RG4**', which encodes */N/* deletion parallelized with four phonetic stem changes:

```
define RG4 %^N->0||[m e|[p e]]_"lbraces"* [l|m|n|r|y|w|t|s|p|k] ,,
          t->n||[m |p] e %^N "lbraces"*_ ,,
          p->m||[m|p] e %^N "lbraces"*_ ,,
          s->n y||[m|p] e %^N "lbraces"*_ ,,
          k->n g||[m|p] e %^N "lbraces"*_;
! push defined RG4;
```

This rule actually consists of five rules that are parallelized. The first rule is */N/* deletion, where the nasal symbol */^N/* is omitted if it is preceded by */me/* or */pe/* and followed by either of the phonemes */l/, /m/, /n/, /r/, /y/, /w/, /t/, /s/, /p/*, or */k/*.

The next four rules are the phonetic stem changes, which apply to stem words preceded by the morpheme */me^N/* or */pe^N/*, and whose initial phoneme is one of */t/, /p/, /r/, /s/*, or */k/*. More specifically, the phoneme */t/* will transform into */n/*, */p/* will transform into */m/*, /s/ will transform into */ny/*, and finally the phoneme */k/* will transform into */ng/*.

All morphophonemic rules that implement the various processes are composed into one large rule,

from which the morphophonemic transducer network can be constructed.

The morphophonemic rule that applies to the word *me^Npukuli* used as an example in Section 5.1 would be the above "*/N/* deletion parallelized with four phonetic stem changes", because the word *me^Npukuli* fulfils the requirements of that rule. As can be seen from the implementation, to get the "*/N/* deletion process" and "*/p/* replacement with */m/* process" ("*/p/* replacement with */m/* process" is one of those four phonetic rules), in that word there must appear the phoneme */^N/* preceded by */me/* or */pe/* and followed by phoneme */l/, /m/, /n/, /r/, /y/, /w/, /t/, /s/, /p/,* or */k/* and it also must have the phoneme */t/, /p/, /r/, /s/,* or */k/* preceded by the morpheme */me^N/* or */pe^N/*. Since the word *me^Npukuli* fulfils these requirements, */^N/* will be omitted and */p/* will transform into */m/* so that the word *me^Npukuli* will transform into the word *memukuli*. With the word *memukuli* being generated, the process is finished and the word *memukuli* is recognised as a valid word in Indonesian.

6 Evaluation

To assess our implemented system, we ran some test cases in the form of words extracted from an electronic version of the *Kamus Besar Bahasa Indonesia*[3]. We tested the implementations of the morphotactic and morphophonemic rules separately. To evaluate the ability of the analyser to both accept valid forms and reject invalid forms, the test cases included both valid and invalid morpheme combinations. Executing all of the test cases, we obtained the results shown in Table 1, which presents the results of morphotactic test cases, and Table 2, which presents the results of morphophonemic test cases. The 'Analysis' column displays the results of test cases where the surface form of an Indonesian word was provided as input, and our system was tasked with parsing the morphological structure. For example, given the word *memukul*, our system should return *pukul*+VERB+AV. On the other hand, the 'Synthesis' column concerns the opposite situation, i.e. test cases where the input is the string of morphological tags, and our system was tasked with generating the fully inflected form.

Result	Analysis	Synthesis	Total
1. Correct result	103	43	146
2. Several results, including correct	46	106	152
3. Incorrect result	3	3	6
Total	**152**	**152**	**308**

Table 1. Results of morphotactic test cases

Result	Analysis	Synthesis	Total
1. Correct result	51	21	72
2. Several results, including correct	6	36	42
3. Incorrect result	1	1	2
Total	**58**	**58**	**116**

Table 2. Results of morphophonemic test cases

We classify the test case results into three categories. The first category indicates that our system returned exactly one correct analysis or synthesis for a valid test case, or does not produce anything for an invalid test case. The second category is that, when given a valid test case, the system produces several answers, one of which is the desired result. Finally, the last category is seen when the system fails to analyse or synthesize a valid test case, or incorrectly produces an answer for an invalid test case.

From the tables, we can observe that the results for analysis are more accurate than for synthesis, where the system tends to produce more than one result. For example, our system correctly provides the single analysis for *memukul*. However, when trying to synthesize *pukul*+VERB+AV, it produces other alternatives, e.g. *memukulkan*, *memperpukuli*, *mengepukulkan*, etc. This suggests the need for a more refined set of morphological feature tags.

7 Summary

We have presented the design of an Indonesian morphological analyser that provides a detailed analysis of its rich affixation process using the two-level morphology approach, implemented using `xfst` and `lexc`. Our approach is able to handle reduplication, a non-concatenative morphological process. Our (not very rigorous) evaluation shows that the implementation is generally able to encode the rules of the various morphological processes.

[3] http://www.pusatbahasa.diknas.go.id/kbbi

References

Adriani, Mirna, Jelita Asian, Bobby Nazief, Seyed Mohammad Tahaghoghi and Hugh Williams. 2007. Stemming Indonesian: A Confix-Stripping Approach. *ACM Transactions on Asian Language Information Processing*, Vol. 6, No. 4.

Alwi, Hasan, Soenjono Sardjowidjojo, Hans Lapoliwa, and Anton Moeliono. 2003. *Tata Bahasa Baku Bahasa Indonesia Edisi Ketiga*. Pusat Bahasa dan Balai Pustaka, Jakarta.

Beesley, Kenneth and Lauri Karttunen. 2000. Finite-State Non-Concatenative Morphotactics. In *Proceedings of SIGPHON-2000*, August 2000, Luxembourg, pp 1-12.

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

Dalrymple, Mary, Maria Liakata, and Lisa Mackie. 2006. Tokenization and Morphological Analysis for Malagasy. In *Computational Linguistics and Chinese Language Processing Vol.11, No.4, December 2006, pp 315-332.*

Gordon, Raymond (ed). 2005. *Ethnologue: Languages of the World 15th edition*. SIL International, Dallas, USA.

Hartono, Hendra. 2002. *Pengembangan Pengurai Morfologi untuk Bahasa Indonesia dengan Model Morfologi Dua Tingkat Berbasiskan PC-KIMMO*. Undergraduate thesis, Faculty of Computer Science, University of Indonesia

Jurafsky, Daniel and James Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Lingusitic, and Speech Recognition*. Prentice Hall, New Jersey.

Koskenniemi, Kimmo. 1983. *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. Publication 11, University of Helsinki, Department of General Linguistics, Helsinki.

Siregar, Neil. 1995. *Pencarian Kata Berimbuhan pada Kamus Besar Bahasa Indonesia dengan menggunakan Algoritma Stemming*. Undergraduate thesis, Faculty of Computer Science, University of Indonesia.

Sneddon, James. 2003. *The Indonesian Language: Its History and Role in Modern Society*. UNSW Press, Sydney, Australia.