

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 in developing an Indonesian morphological analyser that provides a detailed analysis of the rich affixation process. We model Indonesian morphology using the two-level morphology approach, decomposing the process into a set of morphotactic and morphophonemic rules. These rules are implemented as a complex 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 words are built up from smaller units called morphemes, the minimal meaning-bearing units in a language (Jurafsky, 2000). For example, in English, the 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 *-er* and *-s*, examples of affixes, 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 and sub-categorisation frame of the word if it is a verb. This process is called morphological analysis. The value of a morphological analyser is twofold: from a theoretical (linguistic) viewpoint, it is a very useful tool for linguistic modeling and testing certain analyses. On the other hand, from a practical view-

point, it supports many NLP applications such as information retrieval, search engines, and machine translation.

There has been some previous work in developing morphological tools for Indonesian. 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 do not provide any more linguistic information beyond the stem. Hartono (2002) presents an initial version of a morphological analyser developed with PC-KIMMO. Unfortunately, it does not handle reduplication, a key aspect of Indonesian morphology. The morphological analyser that we are developing is designed to be able to handle 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 Indonesian. Finally, section 6 presents some evaluation results of the analyser we have developed.

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 world¹.

Words in Indonesian are built from the roots by means of a variety of morphological operations such as compounding, affixation, and reduplication.

¹ According to *Biro Pusat Statistik*, as of 2004.

Indonesian concatenative morphology regulates how a stem and affixes glue together, while non-concatenative one combines morphemes in more complex ways.

Affixes in Indonesian language can be classified as four categories (Alwi, 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 combination 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. 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*). All regular reduplicated words are recognized in this research.

Indonesian affixation and reduplication are illustrated in the following example. From the stem *main*, we can derive words like *pemain* (by concatenating the prefix *peN-*), *memainkan* (by concatenating the circumfix *meN-kan*), and *mainan-mainan* (by first concatenating the suffix *-an*, then applying full reduplication). Other examples include *bermain-main*, *memain-mainkan*, and *di-main-mainkan*.

Morphology generally makes a distinction between inflectional and derivational processes. In the previous example, the formation of *memainkan* appears to be ‘inflectional’ as the formation does not change the category of the stem, and the formation of *pemain* and *mainan-mainan* is derivational because the derived words are nouns while the stem *main* is a verb. However, the distinction between inflection and derivation, particularly in Indonesian, is not always clear cut. The formation of verbs such as *memainkan* from *main* 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 is not just fusing 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 the 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 language like English, Finnish, Japanese, Russian, and French. It is not only useful to account for various concatenative morphology, but is also able to handle non-concatenative processes, as has been developed in Malagasy tokenization and morphological analysis (Dalrymple, 2006).

4 Design

Our design of 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 lexicon, that word will be analysed by morphotactic and morphophonemic rules. Only after finishing the process in morphotactic and morphophonemic

rules, the result of morphological analyser for that word can be delivered.

The explanation below will explore a little more about 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 since they are already accounted for in the morphotactic rules.

For our initial design, our lexicon is divided into four classes, i.e. verbs, nouns, adjective, 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 order to develop a morphological analyser since the tags will deliver linguistic information that occurs on a word that being analysed.

In our research, the tags 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**, and **+Instrument**.

Tags like **+Verb**, **+Noun**, and **+Adj** give the part of speech information for the non-stems words. On the other hand, **+BareVerb**, **+BareNoun**, **+BareAdj**, and **+BareEtc** give the part of speech information for stems.

Some other tags also give important linguistic information, such as voice, a grammatical category playing a central role in Indonesian syntax. Therefore, tags such as **+AV**, which indicates active voice, **+PASS**, which indicates passive voice, and **+UV**, which is reserved for undergoer voice, are very important.

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**) indicated either it

is an actor or an instrument that being brought with the construction of that word.

4.3 Morphotactic rules

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

In Indonesian, the morphotactic rules 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 (bare 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 (bare verb, noun, adjective or etc) → Noun. Example: *pembelajaran* (*peN*+*ber*+*ajar*). In this example, the word *ajar* is a verb. After merging with *peN-ber-*, the resulting word is a noun.
- *ke*+*ber*+stem (bare 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 (bare 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:

- Pure reduplication without affixation. Example: *buku-buku*. In this example, the word *bu-*

ku-buku, which is a noun, is generated from the word *buku*, which is also a noun.

- Pure 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 checking of additional tags. After finishing all 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 rules that model the changes in the phoneme of the stems, whereas the second group consists of seven rules that model the changes in the phoneme of affixes.

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*.

As seen above, the replacement processes handled by these rules are not yet complete, as we still have to parallelize those rules with some rules in the second group. The seven rules from second group can be seen as follows:

- /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 paralle-

lized, 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.

There are also several revisions made at morphophonemic rules for the sake of reduplication process. For example, the morphophonemic rule ‘/k/ replacement with /ng/’ that parallelized with ‘/N/ deletion’. Before reduplication process being designed, those parallelized rules suppose to be ‘/k/ deletion’ and ‘/N/ replacement with /ng/’. But when the reduplication process was being designed, it was discovered that these rules must be revised. For example, the word *mengotak-ngotakkan* will not be analysed properly if the morphological analyser uses ‘/k/ deletion’ and ‘/N/ replacement with /ng/’ at the morphophonemic design. 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’. With this rule, words such as *mengotak-ngotakkan* can be properly generated.

5 Implementation

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

The explanation below will present the implementation of morphotactic rules, morphophonemic rules, and a brief example about how a word is being analysed by this Indonesian morphological analyser.

5.1 Morphotactic rules implementation

The implementation of morphotactic rules can generally can be illustrated as a flow of process, illustrated in Figure 1.

The morphotactics process will move through those paths. First, it begins with root where this part defines the next continuation classes. From root, it moves to Preprefiks atau prefiks where this

part consists of all variations of preprefixes and prefixes. Preprefixes are necessary to encode the possible morphological variations that take two prefixes, e.g. *memper-*, *diper-*, etc. After that step, the process will continue to Stem section. From this section, the stem of that word is received. In the middle of this flow of process, there are Redup1 and Redup2 section that are responsible in reduplication cases. Redup1 is responsible for partial and affixed reduplications, while Redup2 is responsible for pure reduplication. The Sufiks part defines all variations of suffixes. The last part Pemeriksaan Akhir Tags is responsible for handling all of the special tags.

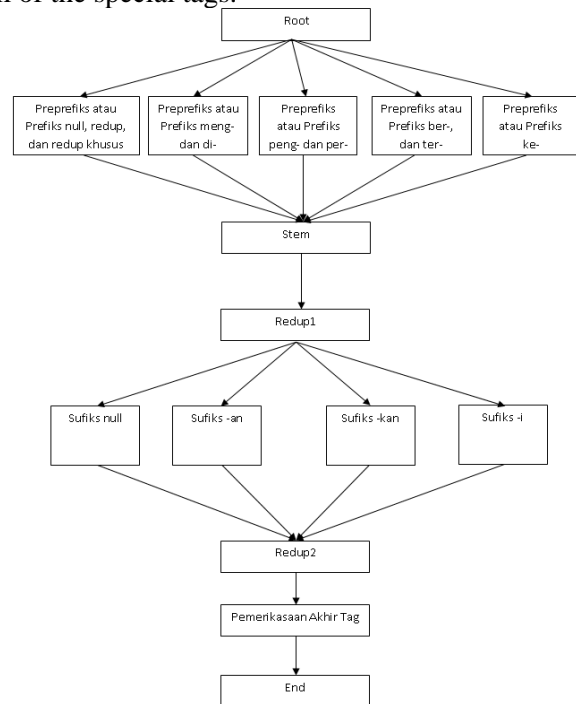


Figure 1. Illustration of Process Flow

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 only valid paths of the network may be traversed. There are three flag diacritics used in our model: positive setting (**@P.featt.val@**), required test (**@R.featt.val@**), and disallow test (**@D.featt.val@**). To exhibit the usage of flag diacritics in our morphological analyser, two examples will be delivered. The first example is the valid word *permainan*, while the second example is the invalid word *memainkan*.

For the first example, the word *permainan* will follow the path highlighted in Figure 2.

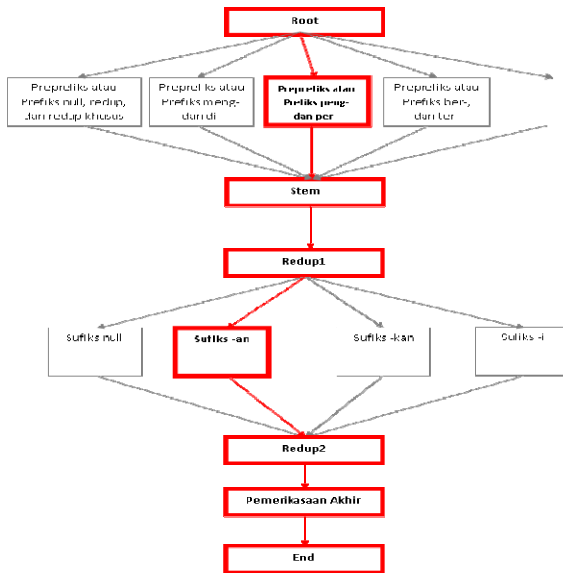


Figure 2. The path for *permainan*

During the first stage that handles prefixes, the network parses the prefix *per-* in *permainan* and positively sets the flag diacritic for the feature **PREF** with value **per**. Subsequently, at the stem stage, the network also positively sets the flag diacritic for the feature **STEM** with value **verb**.

Various required and disallow test flags will be called at the *Redup1*, *Sufiks-an*, *Redup2*, and *Pemeriksaan Akhir* stages. For this example, at each stage there is always at least one path that satisfies the constraints imposed by the flag diacritics. Thus, this word is successfully parsed. For example, at the suffix stage, there is a specific path for the morphological process *per-an* with the following flags: @D.PREPREF@, @D.REDUP@, @R.PREF.per@, @R.STEM.verb@, and @P.SUFF.an@. The first three flags require that the prefix must simply be *per-* (note the disallowing of any positive value for **PREPREF**, e.g. *memper-*, *diper-*) and without any reduplication, whereas the fourth tag requires the stem must be a verb. Finally, should all these flags be satisfied, the network positively sets the feature **SUFF** with value **an**, which in turn will be tested by the subsequent stage, *Pemeriksaan Akhir*.

On the other hand, the second example, i.e. the word *memainkan*, follows the path in Figure 3. As shown, the path stops at the *Sufiks-an* stage. That kind of thing happens because there are required and disallows flags that prevent the invalid word

memainkan to continue the process. It couldn't continue the process since this case only have positive (re)setting flag diacritics for feature **PREF** with value **meN** and feature **STEM** with value **verb** while to pass the Sufiks section, we need flag diacritic for feature **PREF** with value **per-**, **ber-**, **ter-**, **ke-**, or **null**. With that required and disallow flags, the invalid words can be blocked.



Figure 3. The path for *memainkan*

5.2 Morphophonemic rules implementation

The implementation of morphophonemic rules is a little bit different from the implementation of morphotactic rules. In morphotactic rules, there are several steps that can be illustrated as a flow of process. In the other hand, the implementation of morphophonemic rules generally implies the rules itself. Each rule is defined as a replacement rule that will be collaborating with other rules through the method of composition or parallelization.

The implementation of those rules can be seen below:

- /N/ replacement with /n/

```
define R1 %^N->n|[m e]_ "braces"* [d|c|j|s y]);
! push defined R1;
define R2 %^N->n|[p e]_ "braces"* [d|c|j];
! push defined R2;
```

- /N/ replacement with /m/

```
define R7 %^N->m|[m e]_|[p e]_ "braces"* [b|f];
! push defined R7;
```

- Special /N/ deletion

```
define R9 r %^N->0|[p e]_ "lbraces"* [r ["Cons"+ e r]];
! push defined R9;
```

- /N/ replacement with /nge/

```
define R10 %^N->n g e|[m e]_ "lbraces"* "Cons"* "Vow" "Cons"* .#;
! push defined R10;
```

- /N/ deletion parallelized with four phonetic changes' rules

```
define RG4 %^N->0|[m e][p e]_ "lbraces"* [|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;
```

- /r/ deletion

```
define R3 r %+->0|[b e]_ "lbraces"* [r ["Cons"+ e r]];
! push defined R3;
define R4 r %+->0|[t e]_ "lbraces"* r;
! push defined R4;
define R6 r %+->0|[p e]_ "lbraces"* r;
! push defined R4b;
```

- Reduplication special form

```
define lbraces ["^|'|"|"{"|"}|"]|"^|2|^"];
! push defined lbraces;
```

The six rules implementation above will be composed into one rule. From that one rule, the network of morphophonemic rules can be obtained.

Reduplication special form is simply a special sign so that those with a reduplication form can get the same treatment as the affixation form.

5.3 Reduplication Process

To handle the reduplication process we utilize compile-replace feature in xfst. This feature enables us to duplicate certain words by simply give a particular sign at those words. That particular sign is being given in morphotactics process.

If we look back into the flow of process at morphotactic implementation, the treatment for reduplication can be found at *Preprefiks dan Prefiks*, *Redup1*, and *Redup2* sections. In order to show the treatment of reduplication in this morphological analyser, the example with the word *bermain-main* will be delivered.

The word *bermain-main* will follow the path in Figure 4. At *Preprefiks dan Prefiks* section, it receives the first reduplication sign and positive (re)setting for feature **PREF** with value **redup**. Since the case is partial reduplication, then the second reduplication sign will be given at *Redup1* section. *Redup1* section handles the partial and affixed reduplication cases while *Redup2* section

handles the full reduplication cases. At *Redup1* section there are required and disallow tests that can be fulfilled if that case have received the flag for feature **PREF** with value **redup**. After finishing the morphotactic and morphophonemic process, the word *main* that has been given two the reduplication signs will be processed using the compile-replace feature. After that compile-replace process completed, the word *main* will be duplicated and will becomes *main-main*. Since it has been given the prefix *ber-* at *Preprefiks dan prefiks* section in morphotactic process, the word *bermain-main* will be produced.

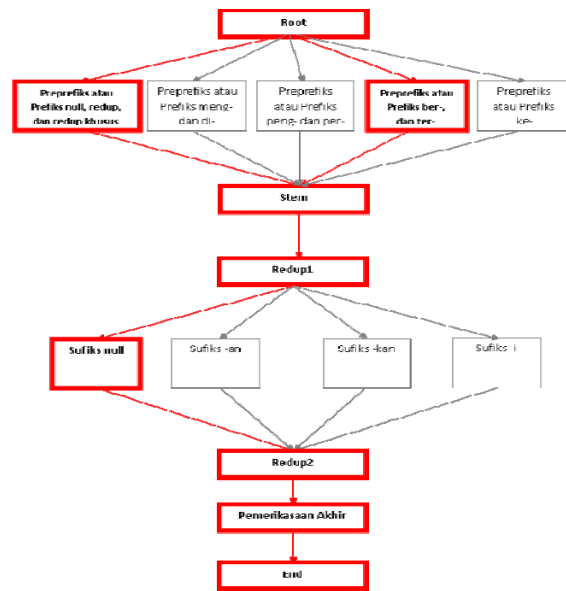


Figure 4. The path for *bermain-main*

5.4 Example process

In order to show the actual process of morphotactic and morphophonemic implementation, we will see how the morphological analyser works at the example below. The case we choose to become our example is:

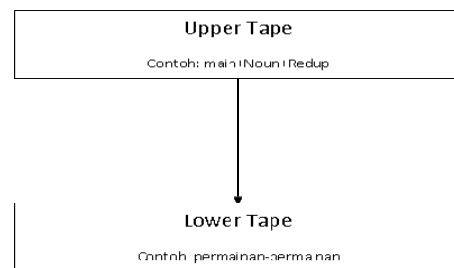


Figure 5. The example for the whole process

That is the example from the pure reduplication case. The program running for this example will follow this path:

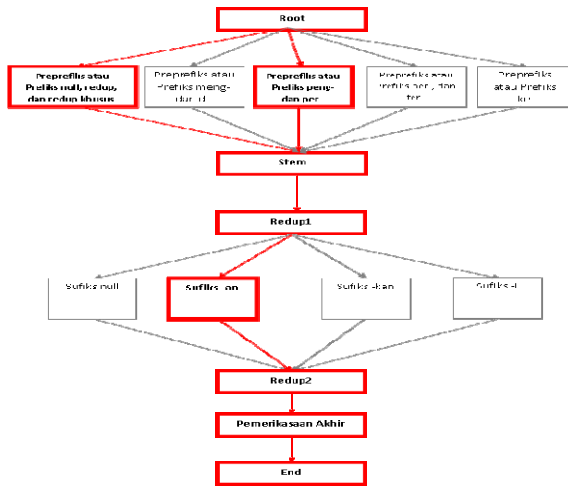


Figure 6. The path for *permainan-permainan*

At the *Root* section, the program moves from root class to *prefiks* class and then from *prefiks* class to *preprefiks redup* continuation class. At this part we move on to *Preprefiks dan prefiks* section.

At *Preprefiks dan prefiks* section, the reduplication sign and positive (re)setting for feature **PRE-PREF** with value **redup** is being given. From *pre-prefiks redup* class the program continues to *prefiks per-* continuation class. At this point, the prefix *per-* and positive (re)setting for feature **PREF** with value **per-** is being given. After that process, the program is heading to stems class at *Stem* section.

In the stems class, the stem main is being given. From this class, the program will heading forward to *POSFlag* class. In *POSFlag* class, positive (re)setting for feature **STEM** with value **verb** is being given. After this, the program moves to *Redup1* section.

At *Redup1* section, the program will just moves on to *Sufiks -an* class at *Sufiks* section since *Redup1* is build to handle the partial and affixed reduplication. At *Sufiks -an* class, the program receives the suffix *-an* and follow the path according to the flag diacritics that this program already got. From *Sufiks* section, it continues to *Redup2* section.

At *Redup2* section, the second flag diacritics sign is given. After this part, the program will continue to *Pemeriksaan Akhir* and finish the morphotactic flow.

There are no rule in morphophonemic process that applied to this case, so the program will move on to compile-replace process.

At compile-replace process, the word *permainan* will be reduplicated. After this process, the word *permainan-permainan* is produced and that is the end of morphological analyser process.

6 Evaluation

To prove performance of Indonesian morphological analyser, we have run some test cases in form of words that extracted from online version of Kamus Besar Bahasa Indonesia². We tested morphotactics and morphophonemics each other separately. The test cases include both valid and invalid morphemes. Executing all test cases, we got the result as shown in table below.

Result Analysis	Apply Up	Apply Down	Total of Process
Correct result	154	64	218
Several results, including correct	52	142	194
Incorrect result	4	4	8
Sum	210	210	420

Table 1. Result of executing test cases

We evaluate the result into three categories. Correct result means that our Indonesian morphological analyser give the appropriate answers. It returns exactly one right answer for word with valid morphemes and does not produce anything whether the word containing invalid morphemes. However, sometimes the analyser produces more than one answer or even there is no suitable answer found.

7 Summary

We have presented the design of an Indonesian morphological analyser that provides a detailed analysis of the 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 well.

² <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. 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 Linguistic, 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.