A Computational Description of Japanese Morphology

Alan W Black*
Dept of Artificial Intelligence, University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN
awb@ed.ac.uk

March 1990
— DRAFT —

Abstract

This paper gives a computational description of Japanese morphology. The description is given in terms of the Edinburgh/Cambridge Dictionary and Morphological Analyser system (MAP) which is part of the Alvey Natural Language Tools. This paper exists for two reasons firstly as a description of Japanese Morphology but also as a description of how more complex morphological systems can be described within the analyser system.

1 Background

The Alvey Natural Language Tools project consists of a GPSG parser, a grammar development environment and a morphological analyser and lexicon component, as well as a large grammar and lexicon for English. The morphological component (MAP) is free standing and is designed as a general tool for building morphological analysers and lexicons for natural language tools ([Ritchie87a], [Ritchie87b]). The system is designed so that a lexicon can be described whose entries consist of morphemes (roots, prefixes, suffixes etc). Then rules can be written about how these morphemes can be joined together to form words. There are two levels of rule formalism used in the description of a morphological analyser. Firstly the lower level termed "morphographemic" describes the

*The author is currently supported by an SERC studentship (award number 89313458), though much of the original description was developed while funded by the Japanese Ministry of Education as a research student in the Dept. of Electrical Engineering, Kyoto University, Japan during 1988. I would like to thank (** all you nice people who read this and give me comments **)

small changes that occur at the concatenation of different morphemes. This is done using spelling rules based on the "two-level morphology" paradigm of [Koskenniemi83a]. The second level within the MAP system is the "morpho-syntax" level which describes which morphemes can be concatenated this is done using a word grammar. The rules of which use feature structures and unification as their base.

In addition to the actual analyser a comprehensive set of lexical rules are available for describing lexical entries in a succinct way.

The original analyser system was designed primarily for use in English language processing systems, but has been used for describing some other languages (as well as being used for describing phonology). Currently the most comprehensive description is of English [Ritchie87c].

This paper is probably too detailed to be read as a whole, unless the reader has a good knowledge of Japanese morphology and the MAP system: the explicit rule descriptions are probably too abstract to follow, these sections may be skipped without loss of continuity.

The rest of this paper consists of a brief background of the Japanese language, then each section of the description is described: morphographemics, morpho-syntax and the actual lexical entries. In each section a description of some the interesting points are made followed by a detailed description of the actual rules used. After that some example analyses are given and finally there is a discussion of the whole system.

2 Japanese Language

Some background in the Japanese language is probably necessary to understand some of this description. The morphological analyser system deals primarily with orthography, its basic function is to take words and provide an analysis of them. In Japanese we immediately have a problem here. The Japanese language is written basically using three different scripts. The exact usage is difficult to define but can roughly be characterized as follows: *hiragana* (a phonetic alphabet) for function words, inflections and common words; *katakana* (a phonetic alphabet with a one to one mapping to *hiragana*) for foreign words, pronunciation; and *kanji* (Chinese characters, ideographs) for major words (nouns, verbs, adjectives etc). Also the roman alphabet can be used for initials and some foreign words. Text is written using a mixture of these alphabets without spaces between words.

Kanji can be written as *hiragana* representing the way those characters are pronounced (though there may be more than one way to pronounce the *kanji* and hence more than one way to map them to *hiragana*). In turn there is a mapping from *kana* (*hiragana* or *katakana*) to roman characters, called *romaji*. There are in fact a number of mappings. The two major ones are the Hepburn system which results in a romanized form which is easy for an English speaker to pronounce correctly, and *kunreishiki* which is more methodical and is recognised as the official style by the Japanese Government.

Because actual typescript (*kana* or *kanji*) is not directly related to morphology and that the work was originally done on a machine which did not support Japanese characters this description was done using *romaji* (using the Hepburn system).

The description here deals mainly with verb and adjective inflection because it is the easiest to describe. There is also a rich morphology in compound nouns and verbs but that is not dealt with here.

One of the first stages required in developing a lexicon and morphological analyser is deciding on what parts of the language should be morphemes. Because the Japanese language does not separate words with white space like English other possibilities exist. The standard item of analysis in Japanese is the *bunsetsu*. This can be thought of as representing a number of words (i.e a phrase). This description, however deals with individual words thus it expects the sentence to be pre-segmented.

3 Morphographemics

This section describes the "morphographemics" of Japanese morphology. That is the small changes that occur when two morphemes are combined. For example in English there is a phenomena of *e-deletion* when the suffix *+ed* is added to a word already ending in *e* (e.g. *move +ed* → *moved*).

Let us first deal with verb inflections. Japanese verbs fall basically into two classes. One can be termed *suru* verbs which are basically verbs formed from *kanji* noun compounds compounded with the basic verb *suru* (*suru* means "to do"). Thus a vast class of Japanese verbs inflect in exactly the same way. The other class is native verbs. These have a fairly rich but describable inflection schema. The differences can be determined by the last *kana* character (one or two characters in *romaji*) of the verb in base form. This second class can further be classified as *-RU*-verbs, which can be thought of as regular, and the rest which have their own inflectional schemas. The possible endings are

> *-U*, *-KU*, *-SU*, *-TSU*, *-NU*, *-BU*. *-MU* and *-RU*.

The last in this list is interesting in that we can have two verbs whose dictionary form is the same but one is a regular *-RU* verb and the other is not and hence inflects differently. (e.g. *iru* (to live) is regular, while *iru* (to need) is not – these would normally be distinguished by different *kanji*, but not always). To distinguish between these two apparently similar verbs we must mark the citation form in the lexical entry accordingly (as does any normal Japanese dictionary). In fact because we will be writting spelling rules that describe changes to characters we will use a technique of writing all "changeable" characters in upper case. That is the lexical entry for *miru* (to see) will be `miRU` as the "`RU`" can be changed (deleted) when inflected. And our ambiguous *iru* is distinguished as `iRU` (to live) and `irU` (to need). This makes the spelling rules easier to write as they can now be written in terms of these special characters as context rather than something more complex.

Let us now look at some examples of inflection and how the changes that occur can be described using the spelling rules. The present form of verbs can be in dictionary form (sometimes called plain form) or in polite form (sometimes called *masu*-form)[1]. The polite form is formed by adding the suffix *+masu* to the dictionary form. In regular verbs *-RU* is deleted and *+masu* added while in the others basically the final `U` is deleted and replaced with an `i` (this may cause other changes too). For example

`miRU`	`+masu`	becomes	`mimasu`	see, polite
`kakU`	`+masu`	becomes	`kakimasu`	read, polite
`dasU`	`+masu`	becomes	`dashimasu`	go out, polite
`yomU`	`+masu`	becomes	`yomimasu`	read, polite

A spelling rule specifies a matching of a pair, a surface character and a lexical character, that is valid within some context. Thus we need to look at the actual pairwise matching of characters on the surface and lexical strings. (The character `0` is treated as a null and although appears in rules it does not need to be explicitly included in words (or citation forms)).

```
lexical string: k a k U + m a s u
surface string: k a k i 0 m a s u
gloss:          to write  polite

lexical string: y o m U + m a s u
surface string: y o m i 0 m a s u
gloss:          to read   polite
```

Unless constrained by a rule, surface characters will match lexical characters that are the same. In all other cases it is necessary to write either a rule or explicitly state that a pair may appear in any context.

If we look at the above two examples we see that we need a rule to deal with a lexical `U` matching a surface `i`. Such a rule is

```
U-to-i
  U:i <=> {k:k m:m} --- < +:0 m:m >
```

This rule states that the pair `U:i` can (and in fact *must*) appear when there is either a `k:k` or a `m:m` to its left and a `+:0 m:m` to its right. {} denote alternatives, and `<>` denote that all must appear (in order).

[1] There is also a more polite form of most verbs used in *keigo* (polite speech) which are often completely differents roots, these are not dealt with here

Of course this only deals with this particular pair, and even then it is not complete. The above phenomena deals with, effectively, a one character change thus needing only one rule to describe it. But what happens when there are two character changes. For example when we add suffixes to regular verbs like `miRU` the `R` and the `U` are deleted.

lexical string:	`m i R U +`	`m a s u`
surface string:	`m i 0 0 0`	`m a s u`
gloss:	to see	polite

lexical string:	`m i R U +`	`t a`
surface string:	`m i 0 0 0`	`t a`
gloss:	to see	past tense

lexical string:	`m i R U +`	`n a i`
surface string:	`m i 0 0 0`	`n a i`
gloss:	to see	negative

To describe this we need two separate rules. One to describe `R:0` pair and another to allow `U:0`.

```
R-to-null
   R:0 <=> =:= --- <U:0 +:0>
U-to-null
   U:0 <=> R:0 --- +:0
```

The `=` is a symbol that represents the set of all characters. The `=:=` is required in the first rule as null left (or right) contexts are not allowed in the formalism. Thus we are using `=:=` to mean “any pair”. Of course `U`-deletion is also needed in other cases so we have to modify this rule as we add new cases.

The solutions to all these are perhaps tedious but not hard. A large number of rules are needed to deal with all cases (and a significant use of the spelling rule debugger to check for inconsistencies).

It should be explained that our technique of using upper case to denote “changeable” characters is used not only at the end of verbs (and adjectives) but also at the end of suffixes themselves. Actually *+masu* inflects (morphographemically) in a similar way to *-SU* verbs so the citation form is actually `+masU`.

Also another use of the upper and lower case distinction for separating apparently similar contexts is used with the `+Ta` suffix (past tense) and `+taI` suffix (want). These cause different inflections of verb stems. It would probably be possible not to use this technique here if the contexts we extended to include the lexical `I` (which can sometimes be realised as a `k` on the surface). This could also be solved by having a different morpheme boundary marker, here we use “`+`” before suffixes we could distinguish *+tai* (want) and *+ta* (past) as `+tai` and `#ta`. It is unclear is either technique is better.

Another place where the case distinction is used (more validly ??) is in adjectives. There are two type adjectives, one always ending in "`i`" has a rich inflectional form and the which does not inflect and is typified by requiring the *na* particle before nouns (hence they are sometimes called *na*-adjectives). Some *na*-adjectives also end in "`i`" so to distinguish between them we use `I` for inflecting adjectives and `i` otherwise, e.g. `akaI` (red) and `kirei` (pretty).

The Spelling Rules

The above gives the general view of how these rules are written now we shall go through each rule in turn with comments.

A spelling rule description consists of a number of set declarations. Sets can be used within spelling rules their interpretation is actually subtle. They do not represent all possible values that are in the set but only those `pairs` which appear explicitly elsewhere in the description.

First we must define the surface alphabet (`0` is assumed in both the surface and lexical alphabet).

```
SurfaceAlphabet
        { a i u e o k g s z j t c d n h f b p m y r w}
SurfaceSets
        = is { a i u e o k g s z j t c d n h f b p m y r w 0 }
        V is {a i u e o}
        C is {k g s z j t c d n h f b p m y r w}
        BNM is {b n m}
```

Likewise for the lexical alphabet and sets

```
LexicalAlphabet
        { a i u e o k g s z j t c d n h f b p m y r w +
          U R I T N}
LexicalSets
        = is { a i u e o k g s z j t c d n h f b p m y r w + 0}
        V is {a i u e o}
        C is {k g s z j t c d n h f b p m y r w}
        BNM is {b n m}
        BM is {b m}
        GK is {g k}
        tT is {t T}
```

All pairs that are valid must either be one to one matches, appear somewhere in a rule or be explicitly declared as a default pair.

```
DefaultPairs
   0:" " U:u b:n m:n k:0 g:0 I:i
```

Because each rule deals with a single character pair but some phenomena affect more that one pair, rules may contain contexts which describe different phenomena thus making the rules even more difficult to read. The first rule deals with changing the final `U` in verbs to an `i` when conjoined with `+masU` (polite), `+yasuI` (derivational inflection changing verbs to adjectives, meaning "easy to"), `+nikuI` (as before meaning "difficult to"), +taI (want to).

```
U-i
 U:i <=>
     {k:= g:=} ---
       < +:0 {tT:= m:m y:y <n:n i:i>}> ;; kakU+Te -> ka0i0te
  or <s:s 0:h> ---
       < +:0 {tT:t m:m y:y <n:n i:i>}> ;; hanasU+Te -> hanashite
  or <t:c s:h> ---
       <+:0 {t:t m:m y:y <n:n i:i>}> ;; matsU+masU -> machi0masu
  or {r:r V:V} ---                   ;; norU+masU -> nori0masu
       <+:0 {t:t m:m y:y <n:n i:i>}> ;; kaU+masU -> kai0masU
  or {b:b m:m} ---
       <+:0 {t:t m:m y:y <n:n i:i>}> ;; asobU+masU -> asobi0masu
  or n:n ---
       <+:0 {t:t m:m y:y <n:n i:i>}> ;; shinU+masU -> shini0+masu
```

`U` at end of verbs changes to an `a` with suffix `+naI` (negative), `+reRU` (passive) and `+seRU` (causative).

```
U-a
   U:a <=> C:C --- < +:0 {<n:n a:a>    ;; yomU+naI -> yoma0nai
                         <{r:r s:s} e:e >}>
    or <V:V 0:w> --- < +:0 {<n:n a:a>  ;; ka0U+naI -> kawa0nai
                          <{r:r s:s} e:e >}>
    or R:= --- <+:0  <{r:r s:s} e:e>>  ;; miRU+reRU -> mira0reru
```

`-RU` is realised as `ra` (in passives) and `sa` (in causatives). Also because we have to restrict `R:r` for this case we need the second clause to deal with the simple case of `R:r` when we match `miRU` → `miru`

```
R-r
   R:r <=> =:= --- <U:a +:0 r:r>      ;; miRU+reRU -> mira0reru
        or =:= --- U:u                ;; miRU -> miru


R-s
   R:s <=> =:= --- <U:a +:0 s:s>      ;; miRU+seRU -> misa0seru
```

The `k` in `-kU` verbs (or `g` in `-gU` verbs) is deleted when the suffixes `+Ta` (past) or `+Te` are added. With `-gU` verbs the `t` becomes voiced so we use an `=` in the `T:=` pair in the right hand context to cope with both realisations. Examples are `kakU+Te` → `kaite` (writing) and `oyogU+Te` → `oyoide` (swimming).

```
gku-tae-1
   GK:0 <=> =:= --- <U:= +:0 T:= {a:a e:e}>
```

H-insertion is necessary when `-SU` verbs change their `U` to an `i`. This is necessary only because we are using the Hepburn system of romanisation.

```
su-tae-1
   0:h <=> s:s --- <U:i +:0>   ;; hanas0U+ta -> hanashita
```

Similarly because of the romanisation method used we have to match `tsU` to `chi` The `t:c` and `s:h` matches are done in separate rules.

```
t-c
   t:c <=> =:= --- <s:h U:i +:0>
s-h
   s:h <=> t:c --- <U:i +:0>
```

The next two rules deal with verbs ending in `-NU`, `-BU` and `-MU`. These change to `n` with suffixes `+Ta` (past) and `+Te`. The `T`'s in turned become voiced. Examples are `shinU+Ta` → `shinda` (died) and `tobU+Te` → `tonde` (flying). Note that we do not include `n` in the rule pair as it will match the surface `n` by default and we do not wish to restrict normal `n:n` pairs. This is a little confusing and comes from the point fact that kana *nu* changes to kana *n* but we represent these using the same roman character.

```
bmnu-tae-1
   BM:n <=> =:= --- <U:0 +:0 T:= {a:a e:e}>
```

The next rules deal with `U`-deletion in the `-NU`, `-BU` and `-MU` case. This also deals with deletion in the regular `-RU` case as in `miRU+Ta` → `mita` (saw). The final clause is for the suffix `+eRU` (potential)

```
bmnu-tae-2
   U:0 <=> BNM:= --- <+:0 T:= {a:a e:e}>
       or  R:0 --- +:0                ;; tabeRU+Te -> tabe000te
       or C:C --- <+:0 e:e>           ;; yomU+eRU -> yom00eRU
```

Voicing of `T` as `d` occurs in some suffixes. This occurs in the `-NU`, `-BU` and `-MU` case and `-gU` verbs.

```
bmnu-tae-3
   T:d <=> <BNM:= U:0 +:0> ---{a:a e:e} ;; yomU+Ta -> yon00da
        or <g:= U:= +:0> --- {a:a e:e}  ;; oyogU+Te -> oyo0i0de
```

T-insertion occurs when the suffixes `+Ta` and `+Te` are conjoined to `-TSU`, `-rU` and `-U` verbs. Examples are: `matsU+Te` → `matte` (waiting), `narU+ta` → `natta` (became) and `kaU+te` → `katte` (buying). This is done by matching the lexical `U` with a surface `t`. Thus the preceding characters need to be deleted.

```
tsu-tae-1                ;; t-deletion in -TSU verbs
   t:0 <=> =:= --- <s:= U:= +:0 T:t {a:a e:e}>
tsu-tae-2                ;; s-deletion in -TSU verbs
   s:0 <=> t:= --- <U:= +:0 T:t {a:a e:e}>
ru-tae-3                 ;; r-deletion in -rU verbs
   r:0 <=> =:= --- <U:= +:0 T:t {a:a e:e}>

u-t-tae-4
   U:t <=>
      <t:0 s:0> --- <+:0 T:t {a:a e:e}> ;; matsU+Te -> ma00t0te
   or r:0 --- <+:0 T:t {a:a e:e}>       ;; norU+Te -> no00t0te
   or V:V --- <+:0 T:t {a:a e:e}>       ;; kaU+Te -> ka00t0te
```

In regular `-RU` verbs the `R` gets deleted, U-deletion is dealt with above

```
RU-1
   R:0 <=> =:= --- <U:0 +:0>
```

This next rule deals with the perhaps dubious treatment of the polite negative. It is treated as a single morpheme `+N` which causes the `U` in `+masU` to change to an `e`. This is another example of requiring reference to actual morphemes in the spelling rules.

```
PL-neg-1
   U:e <=> s:s --- <+:0 N:n>
```

Now we move on to describe the changes that happen at places other than directly after base verbs, i.e after adjectives and some verb suffixes (`+naI` (negative) and `+taI`(want)). Base adjectives all end in `I`. When the past suffix `+Ta` is added the `I` changes to `kat`, as in `ookiI+tta` → `ookikatta` (big, past tense). When the negative suffix `+naI` or `+Te` is added to adjectives the `I` changes to a `ku`. The other change dealt with here is when the potential is added to adjectives and suffixes ending in `I`. `ker` is inserted before `+eba` (potential, non past), as in `chiisaI+0eba` → `chiisakereba` (if (it is) small). These three phenomena are treated by the same group of rules

```
I-ku-ka-ke-1
   I:k <=> =:= --- < +:u n:n a:a >      ;; akaI+naI -> akakunai
        or =:= --- < +:u T:t e:e >      ;; akaI+Te  -> akakute
        or =:= --- < +:a 0:t T:t a:a >  ;; akaI+0Ta -> akakatta
        or =:= --- < +:e 0:r e:e >      ;; akaI+0eba -> akakereba
```

And for the U-insertion for `akaI+Te` → `akakute` (red, conjunctive form).

```
I-ku-2
   +:u <=> I:k --- < n:n a:a >      ;; akaI+naI -> akakunai
        or I:k --- < T:t e:e >      ;; akaI+Te  -> akakute
```

Then two rules to deal with `kat`-insertion

```
I-ka-2
   +:a <=> I:k --- < 0:t T:t a:a >   ;; akaI+0Ta -> akakatta

I-ka-3
   0:t <=> < I:k +:a > --- < T:t a:a > ;; akaI+0Ta -> akakatta
```

The last two rules deal `ker`-insertion

```
I-ke-2
   +:e <=> I:k --- <0:r e:e>         ;; akaI+0eba -> akakereba

I-ke-3
   0:r <=> < I:k +:e > --- e:e       ;; akaI+0eba -> akakereba
```

Comment

The above rules are complex and are not easy to read (or to write). They use a number of techniques to differentiate between contexts, upper case characters in lexical items, inserting over `+` characters as opposed to inserting over `0`. These are successful only because they are never many cases where this has to be done. However it does suggest a problem with the formalism in that it the description is difficult to read. It should be noted that a description of Finnish is probably more complex so perhaps we should not be too worried. There are alternative formalisms for specifying two level morphology (e.g. [Black87]) but that has not been tried.

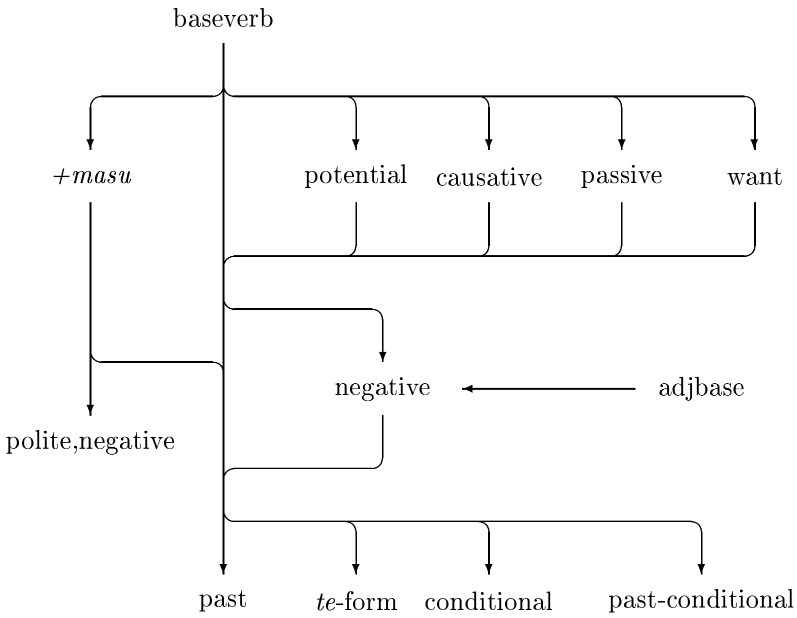
4 Morpho-syntax

Now we have defined the changes that occur when morphemes join, we now have to define *which* morphemes can join.

The descriptions of morphemes in the morpho-syntax section of the analyser are made in terms of categories. Each category consists of a set of feature value pairs, where features can have atomic values or category values thus allowing complex tree structures to be build. The whole formalism is based heavily on GPSG ([Gazdar85]) which a few notable exceptions. The rule structure is standard phrase structure rules rather than ID/LP rules, and there is a different notion of feature passing conventions.

There are an infinite number of ways to try to represent information in the morphemes using feature structures. It is rather strange to develop such a description without also developing a sentence level syntax, as decisions made in the morphology depend heavily on decisions made in the sentence level syntactic description. However an attempt at a general theory-agnostic description is given.

The morpho-syntax section deals with the actual structure of words. In this description that structure is treated very simply. The structure of Japanese morpho-syntax can be viewed as a graph. Each node is a state and each path can be followed by adding a new suffix. Thus, we can view inflections as follows.



This leaves the problem of encoding this in a feature grammar formalism. The simplest way of encoding this is by having a rule which encodes each transition in the graph. For example the rule

```
(Masu_suffixing
       Verb[past +] ->
             Verb[vform base],
             Suffix[past +])
```

says that the `+Ta` suffix can be added to a verb in base form. This method would produce a lot of rules and loose some generality in the description. The first technique we use to reduce the number of rules is using the WSister Feature Convention (via the special feature `STEM`). The WSister convention states that a daughter category may only join with others if its sister categories are "extensions" of the value of its `STEM` feature. This basically means we can define which stems a suffix can take within the suffix itself rather than in an explicit rule (in a way reminiscent of Categorial Grammar).

We identify stems using a feature `STATE`, each constituent will be marked with the feature `STATE` identify the current position in the graph above. The

STEM feature of suffixes will have a category containing the feature STATE. Its value will be the state of which stem it can combine with. For example the suffix +eRU (potential) combines with base verbs so its STEM feature will be

```
(STEM ((STATE BASE)))
```

which states that it can only combine with stems that are marked as BASE. Suffixes that can attach to more than one stem can have a variable value for their state-stem feature. This variable can be declared to range over all and only the states that suffix may conjoin. Also each suffix will have the new state for the combination (it will be copied to the mother category by the WDaughter feature convention, this convention copies features from the right daughter to the mother if they exist else it will copy the defined features from the left daughter, hence it can be used to "replace" features on the stem).

The reason the name STATE is used rather than for example VFORM is that we wish to share states between major classed, a negative verb and a negative adjective although a different class can take the same type of suffix. If we used VFORM (and AFORM) for the feature names representing these forms of words we could not share many of the suffixes without a major change in the design.

This "finite-state" morpho-syntax[2] is very similar to Koskenniemi's original system and how he describes Finnish [Koskenniemi83b] and also Hankamer's description of Turkish ([Hankamer85]). The MAP system offers a much more powerful system of morpho-syntax (features grammars and unification) but, at least for these simple examples, seems not to be needed in Japanese morphology.

In addition to the state information we wish to record in a category which suffixes have been added and how they affect the syntactic description of the word (some of these suffixes are probably normally treated as *semantic* modification but at present we treat them all as syntactic). Each suffix introduces a new feature (e.g. PAST, PASSIVE, CONDITIONAL etc). These features can be thought of being marked as - on all categories (this could be done a the *defaults* mechanism – although they are not). A suffix will add a new feature (with value +) which will be copied to the mother category (because of the WDaughter convention). All other relevant features will be on the stem (not on the suffix) but will still be copied to the mother category (again the WDaughter convention).

These techniques bring the number of necessary rules down to only one. Which simple states that a word can be formed from a non-suffix and a suffix (leaving the conventions and unification to check for validity. The rule looks like (with aliases defined appropriately)

```
(Suffix                    ;; rule name
   NotAffix ->             ;; mother -- aliases for (FIX NOT)
      NotAffix, Suffix )   ;; daughters -- stem and suffix
```

[2] Note that the term "finite-state morpho-syntax" refers to the method used to describe how morphemes are conjoined while "finite-state morphology" is used to describe the whole paradigm of using "spelling rules" (compiled as transducers) to describe morphological changes at morpheme boundaries

A walk-through example may hopefully make things a little less confusing. Lexical entries consists of a five fields: a citation form, a phonetic entry (not used in this description), a syntactic entry (the feature structure), a semantic entry (used to give an English gloss), and one other entry that remains unused. Our entry for `miRU` is

```
(miRU           ;; citation form
 miru           ;; phonology -- ignored
 ((CAT VERB)    ;; the basic category
  (STATE VERBBASE)    ;; the state of this stem
  (FIX NOT))      ;; not an affix
 see            ;; "semantic field" used for English gloss
 nil)           ;; fifth field unused
```

The syntactic part of the entry is actually done with a single alias, and completion rules. The suffix entry

```
(+eRU          ;; citation form
 +eru
 (Suffix        ;; an alias for (FIX SUF)
  (STEM ((STATE BASE)))    ;; what it can combine with
  (STATE POTENTIAL)        ;; the new state after combination
  (POTENTIAL +))           ;; marking to go on the result
 POTENTIAL      ;; "semantic field" used for English gloss
 NIL)           ;; fifth field -- unused.
```

Thus using our only suffixing word grammar rule these two categories can be combined. To produce a mother category of the form

```
((FIX NOT)     ;; from the mother category in the rule
 (CAT VERB)    ;; by WDaughter convention - from left daughter
 (POTENTIAL +) ;; from right daughter (suffix) (WDaughter)
 (STATE POTENTIAL);; new state from right daughter (WDaughter)
)
```

We also use the *MorphologyOnly* class which can be used to distinguish features which are not of interest to the syntax and thus can be removed from the categories generated by morphological analysis. Thus the features `FIX` and `STATE` will be removed in an actual analysis.

The Word Grammar

The actual word grammar contains only one rule but also contains a number of other declarations used in the description. The basic declarations of features and their values, aliases, etc are shared with the lexical entry descriptions. This is done via the “include” facility. The grammar file contains

```
Declarations
   #include "jmorph.dcls"
```

While the file `jmorph.dcls` contains basic feature declarations

```
Feature       CAT     {verb,noun,particle,adj,suffix,conj}
Feature       INTER   {-,+}         ;; has interrogative noun
Feature       POLITE  {-,+}
Feature       PAST    {-,+}
Feature       NEG     {-,+}
Feature       REG     {-,+}         ;; regular / non-regular
Feature       WANT    {-,+}
Feature       POTENTIAL   {-,+}
Feature       PASSIVE     {-,+}
Feature       CONDITIONAL {-,+}
Feature       CAUSATIVE   {-,+}
Feature       PFORM   {wa,ka,ga,no,mo,de,wo,e}
Feature       STEM    category
Feature       FIX     {SUF,PRE,NOT}
Feature       STATE   {BASE,POTENTIAL,CAUSATIVE,PASSIVE,
                          WANT,NOT,POLITE,PAST,AdjBASE,TE,FINAL}
```

Basic aliases

```
Alias         Noun   =   [CAT noun]
Alias         Verb   =   [CAT verb]
Alias         Adj    =   [CAT adj]
Alias         NaAdj  =   [CAT adj]
Alias         Conj   =   [CAT conj]
Alias         Part   =   [CAT particle]
Alias         Suffix =   [FIX SUF]
Alias         NotAffix = [FIX NOT]
```

And variables declarations for the value of `STATE` features in `STEM` categories in suffixes

```
Variable      _ToNai   = {BASE,POTENTIAL,CAUSATIVE,PASSIVE,
                               WANT,AdjBASE}
Variable      _ToPast  = {BASE,POTENTIAL,CAUSATIVE,PASSIVE,
                               WANT,NOT,AdjBASE,POLITE}
Variable      _ToTe    = {BASE,POTENTIAL,CAUSATIVE,PASSIVE,
                               WANT,NOT,AdjBASE}
Variable      _ToIf    = {BASE,POTENTIAL,CAUSATIVE,PASSIVE,
                               WANT,NOT,AdjBASE}
Variable      _ToPolite = {BASE,POTENTIAL,CAUSATIVE,PASSIVE}
Variable      _ToAdj   = {BASE,POTENTIAL,CAUSATIVE,PASSIVE}
```

This is the end of `jmorph.dcls`. The complete word grammar file is as follows

```
Declarations
   #include "jmorph.dcls"

Variable   _top = {verb,noun,adj,particle,conj}
```

The next definition identifies which features are to be subject to the WDaughter feature passing convention. These features will be copied to the mother category from the right daughter (i.e. the suffix) if they exists, or the left daughter (i.e. stem). Thus suffixes can "over-write" feature values on a word.

```
FeatureClass WDaughter = {CAT, NEG, POLITE, REG, STATE,
                          PAST, WANT, POTENTIAL, CAUSATIVE,
                          CONDITIONAL, PASSIVE}
```

The *MorphologyOnly* class describes features which should only exists within the morphology system and not in the external sentence level parsing systems. Thus these features will be deleted from the top category (when CATEGORYFORM style is selected).

```
FeatureClass MorphologyOnly = {STATE,FIX}
```

The next definition states what is a word within the system. Any category which can be found to span the whole input and matches this category definition will be treated as a successful analysis.

```
Top = [CAT _top]
```

Then finally our one grammar rules for suffixing.

```
Grammar
   (Suffix
      NotAffix ->
         NotAffix, Suffix )
```

This description has gone far to try and make itself as succinct as possible and it has probably made it harder to read. This is partly because of a wish to show many facilities that are available in the morphological analyser system. It is not necessary for all descriptions to use so many facilities.

5 Lexical Entries

As already stated, a lexical entry consists of five fields: a citation form, a phonological field (ignored in this description), a syntactic field (feature structure), a semantic field (used for an English gloss) and a "user" field which is not used in this description.

In addition to lexical entries this section includes a number of lexical rules. These can add default features, etc. to basic lexical entries. This means much of the details in lexical entries can be done by rules rather than having to hand specify each entry in entirity.

The lexical rules are of three types, though only one is used in this description. Completion rules add or modify entries. Multiplications rules create new entries (though are not used in this description). Finally there are consistency checks which are used to check the validity of entries, though do not actually do anything active, but are useful when dealing with large lexicons.

The rules are as follows, this simply add features if they are not already present. I.e. the first rule will add the feature `(FIX NOT)` to those entries which do not already have a feature `FIX` (with whatever value).

```
Completion Rules
  Add_NotAffix:
     (_ _ (~(FIX _) _rest) _ _) =>
        (& & ((FIX NOT) _rest) & &)

  Add_INTER_MINUS:
     (_ _ ((CAT noun) ~(INTER _) _rest) _ _) =>
        (& & ((CAT noun) (INTER -) _rest) & &)

  Add_STEM_FIX_NOT:
     (_ _ ((STEM (~(FIX _) _srest)) _rest) _ _) =>
        (& & ((STEM ((FIX NOT) _srest)) _rest) & &)
  Add_REG:
     (_ _ ((CAT verb) ~(FIX SUF) ~(REG _) _rest) _ _) =>
        (& & ((CAT verb) (REG +) _rest) & &)
  Add_STATE:
     (_ _ ((CAT verb) ~(FIX SUF) ~(STATE _) _rest) _ _) =>
        (& & ((CAT verb) (STATE BASE) _rest) & &)
Multiplication Rules
  ;; none
Consistency Checks
  ;; none
```

The lexical entry file first includes the file `jmorph.dcls` as described in the previous section, then includes the lexical rules as above

```
Declarations
   #include "jmorph.dcls"

Rules
   #include "jmorph.rules"
```

The lexicon used in developing this analyser is very small but sufficient for interesting morphology. First some basic roots

```
Entries
;; Nouns
  (bara bara Noun ROSE NIL)
  (daigaku daigaku Noun UNIVERSITY NIL)
  (hon hon Noun BOOK NIL)
  (koen koen Noun PARK NIL)
  (ramen ramen Noun RAMEN NIL)
  (taro Taro Noun Taro NIL)

;; Particles
  (de de Part[PFORM de] DE NIL)
  (e e Part[PFORM e] E NIL)
  (ga ga Part[PFORM ga] GA NIL)
  (ka ka Part[PFORM ka] KA NIL)
  (mo mo Part[PFORM mo] MO NIL)
  (no no Part[PFORM no] NO NIL)
  (wo o Part[PFORM wo] WO NIL)
  (wa wa Part[PFORM wa] WA NIL)

;; Adjectives
  (atarashiI atarashii Adj NEW NIL)
  (benri benri NaAdj CONVENIENT NIL)
  (chiisaI chiisai Adj SMALL NIL)
  (chikaI chikaI Adj NEAR NIL)
  (ii ii (Adj (REG -)) GOOD NIL)
  (nagaI nagai Adj LONG NIL)
  (ookiI ookii Adj BIG NIL)
  (yoI yoi Adj GOOD NIL)

;; Verbs
  (arukU aruku Verb WALK NIL)
  (kakU kaku Verb WRITE NIL)
  (oyogU oyogu Verb SWIM NIL)
  (hanasU hanasu Verb TALK NIL)
  (matsU matsu Verb WAIT NIL)
  (shinU shinu Verb DIE NIL)
  (asobU asobu Verb PLAY NIL)
  (yobU yobu Verb CALL NIL)
  (yomU yomu Verb READ NIL)
  (irU iru Verb NEED NIL)
  (kaU kau Verb BUY NIL)
```

```
(miRU miru Verb SEE NIL)
(iRU iru Verb (BE / LIVE) NIL)
(tabeRU taberu Verb EAT NIL)
(wasureRU wasureru Verb FORGET NIL)
```

Irregular verbs *suru* (to do), *kuru* (to come) and the partially irregular *iku* (to go) have to be hand specified. Where an inflection causes the stem to become regular, we can entry for that case and use the rest of the morphological analyser to deal with the other inflections. For example, the negative form of to do (`shinai`) inflects in all the normal ways of a negative verb so our entry will have the same markings as a stem that has been built from the root and the negative suffix.

```
(suru suru (Verb (REG -)) DO NIL)
(shimasU shimasu (Verb (POLITE +) (STATE POLITE)) DO NIL)
(shinaI shinai (Verb (NEG +) (STATE NOT)) DO NIL)
(sareRU sareru (Verb (PASSIVE +) (STATE PASSIVE)) DO NIL)
(saseRU saseru (Verb (CAUSATIVE +) (STATE CAUSATIVE)) DO NIL)
;; There isn't a potential form, dekiru (to be able to) is used
(shita shita (Verb (PAST +) (STATE FINAL)) DO NIL)
(shitara shitara (Verb (PAST +) (CONDITIONAL +)
                    (STATE FINAL)) DO NIL)
(shikereba shikereba (Verb (CONDITIONAL +)
                    (STATE FINAL)) DO NIL)

(iku iku (Verb (REG -)) GO NIL)
(ikimasU ikimasu (Verb (POLITE +) (STATE POLITE)) GO NIL)
(ikanaI ikanai (Verb (NEG +) (STATE NOT)) GO NIL)
(ikareRU ikareru (Verb (PASSIVE +) (STATE PASSIVE)) GO NIL)
(ikaseRU ikaseru (Verb (CAUSATIVE +) (STATE CAUSATIVE)) GO NIL)
(ikeRU ikeru (Verb (POTENTIAL +) (STATE POTENTIAL)) GO NIL)
(itta itta (Verb (PAST +) (STATE FINAL)) GO NIL)
(ittara ittara (Verb (PAST +) (CONDITIONAL +) (STATE FINAL))
                GO NIL)
(ikereba (Verb (CONDITIONAL +) (STATE FINAL)) GO NIL)

(kuru kuru (Verb (REG -)) COME NIL)
(kimasU kimasu (Verb (POLITE +) (STATE POLITE)) COME NIL)
;; as above for the other inflections of kuru
;; kinaI, kita, kitara, kite, etc.
```

An now the more interesting entries for suffixes.

```
(+eba +eba (Suffix (STEM ((STATE _ToIf)))
            (CONDITIONAL +) (STATE FINAL)) CONDITIONAL NIL)
```

```
(+Tara +tara (Suffix (STEM ((STATE _ToPast)))
            (CONDITIONAL +) (PAST +)
            (STATE FINAL)) CONDITIONAL NIL)
(+Te +te (Suffix (STEM ((STATE _ToTe)))
          (STATE FINAL)) TE NIL)
(+naI +nai (Suffix (STEM ((STATE _ToNai)))
          (NEG +) (STATE NOT)) NEG NIL)
(+Ta +ta (Suffix (STEM ((STATE _ToPast)))
          (PAST +) (STATE FINAL)) PAST NIL)
```

These modify base verbs

```
(+eRU +eru (Suffix (STEM ((STATE BASE)))
            (POTENTIAL +) (STATE POTENTIAL)) POTENTIAL NIL)
(+reRU +reru (Suffix (STEM ((STATE BASE)))
            (PASSIVE +) (STATE PASSIVE)) PASSIVE NIL)
(+seRU +reru (Suffix (STEM ((STATE BASE)))
            (CAUSATIVE +) (STATE CAUSATIVE)) CAUSATIVE NIL)

(+taI +tai (Suffix (STEM ((STATE BASE)))
            (WANT +) (STATE WANT)) WANT NIL)
(+masU +masu (Suffix (STEM ((STATE _ToPolite)))
            (POLITE +) (STATE POLITE)) NIL NIL)
;; this one is pretty weird
(+N +n (Suffix (STEM ((STATE POLITE)))
        (NEG +) (STATE FINAL)) NEG NIL)
```

The next two are derivational suffixes changing verbs into adjectives. Because they are marked with `Adj` (an alias for `(CAT ADJ)` and the feature `CAT` is subject to the WDaughter convention it will be copied up to the mother when combine with a verb rather than the `CAT` feature from the left daughter.

```
(+yasuI +yasui (Suffix (STEM ((STATE _ToAdj)))
                 Adj (STATE AdjBASE)) EASY-TO NIL)
(+nikuI +nikui (Suffix (STEM ((STATE _ToAdj)))
                 Adj (STATE AdjBASE)) HARD-TO NIL)
```

6 Sample Analyses

Now that we have defined our spelling rules, word grammar and lexical entries. We can compile and load them into the morphological analyser system. There are two (major) forms of analysis available from the top level of the analyser system: CATEGORYFORM where the top syntactic category of the analysed word is displayed and WORDSTRUCTURE which will give the whole tree.

The first example is *yomeru* (can read). The whole tree is given.

```
> w yomeru
Looking up word

1 analyses found

analysis : 1

(((cat verb) (reg +) (state potential) (potential +) (fix not))
suffix
   (((bar 0) (reg +) (state base) (cat verb) (fix not))
   entry
      (yomu yomu ((cat verb) (state base) (reg +)
                  (fix not) (bar 0)) read nil))
   (((potential +) (state potential) (bar 0)
     (stem ((fix not) (state base))) (fix suf))
   entry
      (+eru +eru
            ((stem ((fix not) (state base))) (bar 0)
             (state potential) (potential +) (fix suf))
            potential nil)))
```

The segmentation can be seen in the base lexical entries. We can also analyse word to give there syntactic category only (gloss has been added afterwards):

```
> w akakunai             ;; not red
Looking up word

1 analysis found
((cat adj) (past -) (neg +))

> w kakinikukunakatta    ;; was not difficult to write
Looking up word

1 analysis found
((reg +) (cat adj) (neg +) (past +))

> w asobanakereba        ;; if (subject) is not playing
Looking up word

1 analysis found

((cat verb) (reg +) (neg +) (conditional +))
```

The following table shows a large range of analyses. The segmentation is shown and the top category as well as an English gloss.

Word	Segmentation and Syntactic Category	English gloss
miru	miRU	see
	((cat verb) (reg +) (bar 0))	
aruita	arukU+Ta	walked
	((cat verb) (reg +) (past +))	
machitai	matsU+taI	want to wait
	((cat verb) (reg +) (want +))	
oyoide	oyogU+Te	swimming
	((cat verb) (reg +))	
yomaremasu	yomU+reRU+masU	read (passive polite)
	((cat verb) (reg +) (passive +) (polite +))	
hanashitakunakatta	hanasU+taI+naI+Ta	did not want to talk
	((cat verb) (reg +) (want +) (neg +) (past +))	
yonda	yomU+Ta	read (past tense)
	((cat verb) (reg +) (past +))	
	yobU+Ta	called
	((cat verb) (reg +) (past +))	
irimasu	irU+masU	need (polite)
	((cat verb) (reg +) (polite +))	
imasu	iRU+masU	be / live (polite)
	((cat verb) (reg +) (polite +))	
kawanakute	kaU+naI+Te	not buying
	((cat verb) (reg +) (neg +))	
tabemashita	tabeRU—masU+Ta	ate (polite)
	((cat verb) (reg +) (polite +) (past +))	
nagakunakatta	nagaI+naI+Ta	not long (past)
	((cat adj) (neg +) (past +))	
atarashikunakereba	atarashiI+naI+eba	if not new
	((cat adj) (neg +) (conditional))	

Comment

The above morphological analyser and lexicon is more analyser than lexicon. However its can easily (and quickly) be extended. Using a dictionary from a simple *kana*-to-*kanji* input conversion program a far larger lexicon can be built. Note that only the lexical entries have to be added and no changes are necessary to the spelling rules and word grammar. Lexicons of 10,000 morphemes entries are of no problem – though as many of these entries are non-inflecting *kanji* compounds they are not very morphologically interesting.

Adding new affixes is also possible, the above is not an exhaustive list. There is however an issue about what is morphology and what is syntax. For example, some verbs may be thought of as suffixes themselves (e.g. *iru* (be / live) as in *kaite iru* (writing)). It is difficult to know where to stop. A full treatment of Japanese grammar is probably necessary to decide on how to treat these.

It should be noted that there are a number of mistakes in the analysis. For example the `PASSIVE` feature will be copied up on to a mother even though a derivational affix has been added. Thus we can have an adjective labelled passive. This is probably a fault in the morpho-syntax and requires a change to `+yasuI` and `+nikuI` lexical entries.

Also the potential form of `-RU` verbs has the same realisation as the passive form. The spelling rule would need to be modified to deal with this.

A third problem that it is probably the case that the syntactic treatment of suffixes given here (i.e.as different features) is not what a sentence level syntactic grammar would require and hence would need to be modified if it were to be used in a real system.

Certain phenomena have been not been describe but would fit neatly into the description. For example the changes that occur in "count markers" suffix with numbers (`san+hon` $\rightarrow$ `sanbon` (three cylindrical objects)) would be easy to describe. Also the `TSU` to small `TSU` changes that occur in *kanji* compounding.

In essence this description is really only a start at a description but hopefully a good start.

7 Conclusion

In Japanese it may be better to analyse whole phrases as whole phrases rather than individual words as the problem of finding where the words start and stop is basically the same as syntactic analysis. The above description deals only with words but could be extended to take whole phrases and segment them into words. The morphological system does have a facility for doing this and is quite successful at it but this does not include a syntactic parse of the phrase, only a segmentation. It is obvious that these two parts, segmentation and analysis of phrases, should be done together (as they are so in the morphological analyser).

The other point is that although there exists a well defined mapping from *kana/kanji* to *romaji* it is unusual to see Japanese written in roman characters except in foreign dictionaries and in papers such as this. Analyses should (and can) be written using the Japanese native script. This would, however, require a change to the MAP system (though probably not a major one).

Another interesting point is that the issue of computational morphological analysis is, in Japanese, a very useful one which applies to everyday life. Almost all Japanese people use a word processor for producing text. The entry system is somewhat complex and benefits (in the more expensive models) from a simple morphological analyser which will cope with the same changes as the spelling rules describe. It would benefit further from a more complex morphological analyser.

The coverage described here is small in terms of morphemes but fairly large in terms of morphology. However to be of real benefit an analyser would need to deal with *kana* and *kanji* (which is now no longer a problem on a general workstation). The other important, but more difficult requirement, would be an analysis of compound nouns (and possibly verbs) but that would probably require at least a simple semantic analysis of the stems and a fair knowledge of the Japanese language.

There are other analyses of Japanese morphology. Whitelock ([Whitelock88]) deals mainly with the morpho-syntax. His analysis uses unification and a categorial grammar-like analysis, Thus making it very similar to this one. However he relates his analysis to the sentence level in more detail. Another treatment exists in an early implementation of the "two-level" paradigm ([Alam83]). The early KIMMO system did not offer much in the way of morpho-syntax and offered a low level finite state transducer formalism for writing spelling rules. Thus making it significantly harder to write (and read).

The Alvey Natural Language Tools were written with the syntactic theory GPSG very much in mind. A treatment of Japanese syntax has been developed called JPSG ([Gunji87]) which does not itself deal with morphology, however some later work does start at an analysis ([Gunji86]). It seems that it would be an intersting (and quite feasible) task to implement JPSG and its morphological description in the Alvey Natural Language Tools.

As a final point, one of the objects of writing this analysis was to see how good the MAP system is at dealing with languages with morphology significantly more complex than English. There are minor problems with the system but at no time was any change needed to the system. All development took place within the analyser system using the provided debugging systems (primarily the spelling rule debugger), and it was never necessary to "hack" the underlying LISP code. Thus it does suggest the system is suitable for languages other than English.

References

[Alam83] Y. Alam A Two-level Morphological analysis of Japanese *Texas Linguistics Forum*, 22:228–252, 1983.

[Black87] A. Black, G. Ritchie, S. Pullman, and G. Russell. Formalisms for morphographemic description. In *Proceedings of 3rd Conference of the European Chapter of the Association for Computational Linguistics*, pages 11–18, 1987.

[Gazdar85] G. Gazdar, E Klein, F. Pullum and I. Sag *Generalized Phrase Structure Grammar* Blackwell, Oxford, 1985

[Gunji86] T. Gunji. *The JPSG Morphological System (Version 1.5)* unpublished paper, Faculty of Language and Culture, Osaka University, Japan, 1986.

[Gunji87] T. Gunji. *Japanese Phrase Structure Grammar* Reidel, Dordrecht, 1987.

[Hankamer85] J. Hankamer. *Finite State Morphology and Left to Right Phonology* unpublished paper, University of California, Santa Cruz 1985.

[Karttunen83] L. Karttunen. KIMMO: a two-level morphological analyser. *Texas Linguistics Forum*, 22:165–186, 1983.

[Koskenniemi83a] K. Koskenniemi. *Two-level Morphology: a general computational model for word-form recognition and production.* Publication 11, University of Helsinki, Department of General Linguistics, Finland, 1983.

[Koskenniemi83b] K. Koskenniemi. Two-level model for morphological analysis. In *Proceedings of the 8th IJCAI*, pages 683–685, International Joint Conference on Artificial Intelligence, 1983.

[Ritchie87a] G. Ritchie, A. Black, S. Pulman, and G. Russell. *The Edinburgh/ Cambridge Morphological Analyser and Dictionary System: System Description.* Software Paper 11, Dept of Artificial Intelligence, University of Edinburgh, Scotland, 1987.

[Ritchie87b] G. Ritchie, A. Black, S. Pulman, and G. Russell. *The Edinburgh/ Cambridge Morphological Analyser and Dictionary System: User Manual.* Software Paper 10, Dept of Artificial Intelligence, University of Edinburgh, Scotland, 1987.

[Ritchie87c] G. Ritchie, S. Pulman, A. Black, G. Russell. Computational Framework for Lexical Description Journal of Computational Linguistics. 13:290-307, 1987

[Whitelock88] P. Whitelock A Feature-based Categorial Morpho-syntax for Japanese In C.Rohre, U. Reyle, editors *Natural Language Parsing and Linguistic Theories*, Reidel, Dordrecht, 1988