Parsing with Shoebox

David Bevan

1. Introduction

This document describes how to structure your lexical database for parsing with Shoebox for Windows and Macintosh. In Shoebox 2 for MS-DOS, there were two methods of providing parsing information: a parsing database and the conjoined affix parser. With Shoebox 5, neither of these are necessary because the program has a morphological parser which can break off any number of affixes and handle morphophonemic changes to affixes and roots. If you desire, all the necessary information for parsing can be kept in your lexical database.

This document uses English examples with the standard Shoebox interlinear settings and Multi-Dictionary Formatter (MDF) data field markers.

2. Simple Parsing: Roots and Affixes

With Shoebox, it is necessary in the lexicon to specify whether a morpheme is a root, prefix, suffix or infix by the correct use of morpheme break characters (normally hyphens) in the lexeme field. A prefix ***must*** have a final hyphen; a suffix ***must*** have an initial hyphen; a root ***must not*** have a hyphen; an infix ***must*** have an initial and a final hyphen. Note that this means that you cannot have hyphens on bound roots in the lexeme field. However, you can enter roots with hyphens in the \lc Lexical Citation data field.

This is all that's necessary for simple parsing. For example, the lexical entries

\lx	un-	\lx	success	\lx	-ful
\ps	neg	\ps	n	\ps	nadjzr
\ge	OPPOS	\ge	achievement	\ge	ADJZR

will cause unsuccessful to be interlinearized correctly:

unsuccessful		
un-	success	-ful
OPPOS-	achievement	-ADJZR
neg-	n	-nadjzr

Forms with hyphens between the morphemes are also parsed correctly. For example, with

\lx	non-	\lx	read	\lx	-er
\ps	neg	\ps	v	\ps	vnomzr
\ge	not	\ge	look_at_book	\ge	AGENT

in the lexicon, non-reader is parsed as follows:

non-reader		
non-	read	-er
not-	look_at_book	-AGENT
neg-	v	-vnomzr

Note how great an improvement this is over Shoebox 2 where every word had to be parsed individually, or every combination of affixes listed in the conjoined affix parser.

2.1 Infixes

An **infix** is a morpheme embedded within a root or an affix. Some people use the term *infix* to refer to an affix which occurs only between a root and another affix; but for Shoebox that is just another prefix or suffix, not an infix.

English doesn't have infixes, but if

\lx	-int-
\ps	intens
\ge	INTENS

was an English intensive infix, then s**int**uccessful would be interlinearized 'correctly' as

s**int**uccessful		
success	-int-	-ful
achievement	-INTENS-	-ADJZR
n	-intens-	-nadjzr

In the parse, infixes appear before or after the root or stem in which they are found. (Shoebox has an option which lets you choose.) These examples display the infix after. Infixes can be found anywhere in the word, so successf**int**ul would be interlinearized as follows:

successf**int**ul		
success	-ful	-int-
achievement	-ADJZR	-INTENS-
n	-nadjzr	-intens-

2.2 Compound Roots

Words with compound roots will be parsed correctly if both roots are in the lexicon. For example, the following lexical entries

\lx	black	\lx	bird	\lx	-s
\ps	adj	\ps	n	\ps	ninfl
\ge	dark	\ge	flying_creature	\ge	PL

will cause blackbirds to be interlinearized as follows:

blackbirds			
black	-	bird	-s
dark	-	flying_creature	-PL
adj	-	n	-ninfl

Compounds with hyphens, such as sea-green and mother-in-law parse correctly. Shoebox has an option to allow or disallow compound roots.

3. Alternate Forms

If a morpheme has more than one surface form, this can be specified with an **alternate form** (\a) field:

\lx	a	\lx	telephone
\a	an	\a	phone
\ps	art	\ps	n
\ge	INDEF	\ge	transceiver

The parsed form is taken from the \lx field:

an	enormous	phone
a	enormous	telephone
INDEF	very_big	transceiver
art	adj	n

Here are lexical entries for some more morphemes with alternate forms:

\lx	in-	\lx	-s	\lx	not
\a	im-	\a	-es	\a	-n't
\a	il-	\ps	ninfl	\ps	neg
\a	ir-	\ge	PL	\ge	NEG
\ps	neg				
\ge	OPPOS				

and some example interlinear text:

impossible		foxes		faces		haven't	
in-	possible	fox	-s	face	-s	have	-not
OPPOS-	feasible	animal_sp.	-PL	head	-PL	own	NEG
neg-	adj	n	-ninfl	n	-ninfl	v	neg

Note that foxes and faces are parsed correctly due to the presence of fox and face in the lexicon (but not foxe and fac).

Note also that haven't has been analyzed here as a compound.

Alternative forms can have a different (affix) type from the lexeme (e.g. you can have a suffix as an alternative form of a root or prefix).

4. Underlying Forms

If you require that a variant form has its own lexical entry, you can use an **underlying form** (\u) field. Here's an alternative way of treating phone. The parsed output is the same as above.

\lx	phone	\lx	telephone
\u	telephone	\ps	n
		\ge	transceiver

You can enter the morphemic breakdown of a lexical entry in an underlying form field. (This is like the use of the parsing database in Shoebox 2.) For example, here is one way to handle the suppletive verb form went.

```
\lx  went        \lx  go        \lx  -ed
\u   go -ed      \ps  v         \ps  vinfl
                 \ge  proceed   \ge  PAST
```

Here's how it parses:

he	went	
he	go	-ed
3SM	proceed	-PAST
pron	v	-vinfl

Rather than having a separate lexical entry for each irregular form, these forms may be included in the main entry by using an alternate form field ***and*** an underlying form field. Here's another way of handling went along with some other irregular verb forms:

```
\lx  go          \lx  find      \lx  hit
\a   went        \a   found     \u   hit
\u   go -ed      \u   find -ed  \u   hit -ed
\ps  v           \ps  v         \ps  v
\ge  proceed     \ge  locate    \ge  strike
```

An underlying form (\u) field is associated with whichever \lx or \a field precedes it. (If no \u field follows a \lx or \a field, the contents of the \lx field are used as the underlying form.)

Note: When you want to identify portions of underlying forms that are to be parsed as whole units, you ***must*** leave spaces between the morphemes in an underlying form field to distinguish roots, prefixes and suffixes. But if you want the operation to return a portion to the parser to add to the main block being processed, ***no*** spaces are allowed.

Note that hit is ambiguous and so two underlying forms are given to make Shoebox display an Ambiguity Selection dialog box.

Here's how the verbs parse (with the past tense form chosen for hit).

went		found		hit	
go	-ed	find	-ed	hit	-ed
proceed	-PAST	locate	-PAST	strike	-PAST
v	-vinfl	v	-vinfl	v	-vinfl

Note that it doesn't matter to Shoebox whether a form that needs parsing information is included in a main lexical entry as an alternate form or has its own lexical entry (perhaps in a separate database). The choice is yours.

Sometimes, it may be necessary to use an underlying form field for affix sequences — rather like using the conjoined affix parser in Shoebox 2. The affix -ability is an example in English:

```
\lx  -able       \lx  -ity      \lx  -ability
\ps  vadjzr      \ps  anomzr    \u   -able -ity
\ge  ABIL        \ge  NOMZR
```

Here's how readability is parsed:

readability		
read	-able	-ity
look_at_book	-ABIL	-NOMZR
v	-vadjzr	-anomzr

Finally, here are some examples of underlying forms of compounds:

```
\lx  have        \lx  brunch
\a   I've        \u   breakfast lunch
\u   I have
\ps  v
\ge  own
```

which parse like this:

I've		brunch		
I	have	breakfast		lunch
1S	own	morning_meal		noon_meal
pron	v	n		n

4.1 Forced Values — Resolving Ambiguity

Where a morpheme has more than one meaning, it is sometimes helpful in underlying forms to specify which one is required so that Shoebox doesn't display an Ambiguity Selection dialog box. For example, two English suffixes have the form -s, so in the underlying form of the irregular form men, it is worthwhile specifying which one is required. This is done by putting the gloss in curly braces after the morpheme:

```
\lx  -s          \lx  -s          \lx  man
\a   -es         \a   -es         \a   men
\ps  ninfl       \ps  vinfl       \u   man -s{PL}
\ge  PL          \ge  3S          \ps  n
                                  \ge  male_person
```

Then men parses as follows without displaying an Ambiguity Selection dialog box.

```
men
man          -s
male_person  -PL
n            -ninfl
```

For forced glosses to work in this way it is necessary that in the parsed output the gloss line appears immediately after the parsed (morpheme breakdown) line. If the part of speech line was above the gloss line, then the forced value would have to be specified as -s{ninfl} or -s{ninfl}{PL}. Note that multiple forced values are possible in the order of the interlinear lines.

Here's another example, with the ambiguity on the root:

```
\lx  bear          \lx  bear
\ps  n             \a   bore
\ge  animal_sp.    \u   bear{carry} -ed
                   \ps  v
                   \ge  carry
```

With these lexical entries, bore parses like this without displaying an Ambiguity Selection dialog box. (Of course, bore is itself ambiguous, also meaning 'drill a hole'.)

```
bore
bear   -ed
carry  -PAST
v      -vinfl
```

Forced glosses on prefixes must be placed after the hyphen, not before.

4.2 Direct Parsing — Preventing Incorrect Parses

With the following lexical entries,

```
\lx  hop          \lx  hope         \lx  -s
\ps  v ; n        \ps  v            \a   -es
\ge  jump         \ge  expect       \ps  vinfl
                                    \ge  3S
```

the word hopes will be parsed incorrectly:

```
 hopes
*hop   -s
*jump  -3S
 v     -vinfl
```

In parsing, Shoebox resolves much of the possible ambiguity by choosing the parse which cuts off the longest affix. This prevents the user being presented with an Ambiguity Selection dialog box with many possible parses — mostly incorrect — to choose from.

In the case of hopes, this means that -es is cut off.

In situations like this, to get the correct parse it is necessary add parsing information to over-ride the incorrect analysis (an alternative approach is outlined later):

```
\lx  hope
\a   hopes
\u   hope -s{3S}
\ps  v
\ge  expect
```

This forces hopes to be parsed like this:

hopes	
hope	-s
expect	-3S
v	-vinfl

Here's another example involving prefixes. To prevent the false parse of the word demisting in which the prefix demi- gets cut off,

demisting	
*demi-	sting
*half-	hurt
*num-	n

it is necessary to provide an analysis of demist:

\lx	demist	\lx	mist
\u	de- mist{fog}{v}	\ps	n ; v
		\ge	fog

Then the word is correctly parsed:

demisting		
de-	mist	-ing
REVERS-	fog	-PTC
neg-	v	-vinfl

In some cases there is more than one correct parse of a word (or part of word) with morpheme breaks in different places. In this case, it is necessary to provide parsing information for both so that the ambiguity is recognized. Here's an example:

\lx	do	\lx	doe
\a	does	\a	does
\u	do -s{3S}	\u	doe -s{PL}
\ps	v	\ps	n
\ge	perform	\ge	female_deer

Shoebox will display an Ambiguity Selection dialog box for you to choose which analysis you want:

does		does	
do	-s	doe	-s
perform	-3S	female_deer	-PL
v	-vinfl	n	-ninfl

5. Morphophonemics

Simple morphophonemic alternations can be expressed by using alternate forms and underlying forms. Here is how the orthographic rules

y → i / __ +ed *or* y + ed → ied
y → ie / __ +s *or* y + s → ies

are expressed:

\lx	-ed	\x	-s
\a	-d	\a	-es
\a	-ied	\a	-ies
\u	y+ed	\u	y+s
\ps	vinfl	\ps	vinfl
\ge	PAST	\ge	3S

The way this works is that the alternate form field contains the surface form including the whole of the suffix. The underlying form field contains the underlying form of the part of the root (or preceding suffix) that is modified followed by + and the underlying form of the suffix.

Here's how forms of the verbs try and tie are parsed:

tried		tied		tries		ties	
try	-ed	tie	-ed	try	-s	tie	-s
attempt	-PAST	bind	-PAST	attempt	-3S	bind	-3S
v	-vinfl	v	-vinfl	v	-vinfl	v	-vinfl

Here's another example, for the doubling of consonants before the -ed suffix:

```
\lx  -ed
\a   -d
\a   -pped
\u   p+ed
\ps  vinfl
\ge  PAST
```

and here's how hopped parses:

```
hopped
hop   -ed
jump  -PAST
v     -vinfl
```

At present, there is no way of specifying general rules, so this parsing information would have to be provided for each suffix and each consonant that gets doubled.

The approach is equivalent for prefixes. There are very few processes involved in English prefixes, but if we suppose that the rule

sp → p/ dis+ __ *or* dis + sp → disp

as exemplified by the word dispirited was productive, then we would specify it like this:

```
\lx  dis-
\a   disp-
\u   dis+sp
\ps  neg
\ge  OPPOS
```

and dispirited would be parsed as follows:

```
dispirited
dis-      spirit    -ed
OPPOS-    vitality  -possessing
neg-      n         -nadjzr
```

5.1 Context Sensitivity — Preventing Incorrect Parses

Even when there is no morphophonemic process involved, it may still be beneficial to use the morphophonemic notation if an affix's occurrence is phonologically conditioned in order to reduce the number of false parses.

For example, with the allomorphs of the prefix in- specified as follows,

```
\lx  in-
\a   im-
\a   il-
\a   ir-
\ps  neg
\ge  OPPOS
```

words like image, imam, iris and iron will be parsed incorrectly if they don't already have their own lexical entry. For example:

```
 iron
*in-      on
*OPPOS-   upon
*neg-     prep
```

If we use the morphophonemic notation to express the restriction on their occurrence as follows:

```
\lx  in-
\a   imb-
\u   in+b
\a   imm-
\u   in+m
\a   imp-
\u   in+p
\a   ill-
\u   in+l
\a   irr-
\u   in+r
\ps  neg
\ge  OPPOS
```

then the false parses won't occur.

5.2 Ensuring Ambiguity is Recognized

The morphophonemic notation can also be used to achieve the opposite effect of increasing the number of parses. For example, the following specifies that there are two possible parses of word-final -es. Either it is simply -s (as in pushes) or (the additional information) it is e+s (with roots with final -e, as in likes):

\lx -s
\a -es
\a -es
\u e+s
\ps vinfl
\ge 3S

In cases (such as hopes) where roots exist both with and without a final -e (hope and hop), as far as Shoebox is concerned the parse is ambiguous, but without specifying the above (apparently redundant) information Shoebox will always choose the one which cuts of the longer affix. Adding the extra parsing choice makes Shoebox display the Ambiguity Selection dialog box offering a choice (between hop -s and hope -s).

The disadvantage of this approach is that the (false) ambiguity is always displayed unless for each form you add specific parsing information.

On the other hand, for forms like does which really is ambiguous, it makes it unnecessary to provide specific parsing information for either parse.

6. Reduplication

Simple reduplicative processes can be represented in Shoebox. The following example specifies a reduplication of from one to three consonants followed by a vowel at the beginning of the reduplicated root, stem or word. For Shoebox to recognize that this is an entry for reduplication, the \lx field must contain the letters "dup" somewhere. (It may be best for these to occur at the beginning of the field so that all entries for reduplication sort together.) The \a fields are used for specifying the pattern to match using variables defined in the language encoding properties.

\lx dupCV-
\a [cons][vowel]-
\a [cons][cons][vowel]-
\a [cons][cons][cons][vowel]-
\ps intens
\ge very

English doesn't make regular use of reduplication, but if the process above was English, then the following would be correct parses:

bibig		**stro**strong		**bla**black	
dupCV-	big	dupCV-	strong	dupCV-	black
very-	large	very-	powerful	very-	dark
intens-	adj	intens-	adj	intens-	adj

Reduplicative suffixation can also be specified, as can reduplication with fixed letters. For example, the following specifies a reduplication of the final consonant cluster with an intermediate i.

\lx -dupiC
\a -i[cons]
\a -i[cons][cons]
\a -i[cons][cons][cons]
\ps dimin
\ge a_bit

Here's what the parses would look like:

big**ig**		strong**ing**		black**ick**	
big	-dupiC	strong	-dupiC	black	-dupiC
large	-a_bit	powerful	-a_bit	dark	-a_bit
adj	-dimin	adj	-dimin	adj	-dimin

It is possible to specify the type of reduplication in English which copies a syllable, replacing the vowel — usually with i — as in tip-top, tick-tock, criss-cross, flip-flop and wishy-washy. The entry would be as follows:

\lx dupCiC
\a -[cons]i[cons]
\a -[cons]i[cons][cons]
\a -[cons][cons]i[cons]
\ps redup
\ge intens

Reduplication of a whole unit can be specified as follows:

```
\lx  dup
\a   [...]
\ps  redup
\ge  informal
```

By adding hyphens, you can also specify prefix full reduplication or suffix full reduplication.

Here's an English example of full reduplication, which also demonstrates its interaction with suffixation and the fact that reduplicative forms with hyphens are parsed correctly.

```
goody-goody
dup       -  good    -y
informal  -  nice    -FAMIL
redup     -  adj     -familiar
```