Welsh letter-to-sound rules: Rewrite rules and two-level rules compared

Running title: Welsh letter-to-sound rules

Briony Williams

Centre for Speech Technology Research,

University of Edinburgh,

80 South Bridge,

Edinburgh, EH1 1HN.

e-mail: briony@cstr.ed.ac.uk

Full postal address:

Centre for Speech Technology Research,

University of Edinburgh,

80 South Bridge,

Edinburgh, EH1 1HN.

e-mail: briony@cstr.ed.ac.uk

Abstract

In a text-to-speech synthesis system, input words not found in the system's lexicon are passed to letter-to-sound rules, which derive the word's pronunciation. In Welsh, the letter-to-sound rules must be applied in three passes: firstly, to add epenthetic vowels, secondly, to determine stress and vowel location, and thirdly, to perform grapheme-to-phoneme conversion. To begin with, all these letter-to-sound rules were written in the form of context-sensitive rewrite rules, and were evaluated, giving a 96% success rate. The rules for the second pass were then rewritten in the form of two-level rules, using the PCKIMMO software package. The output was identical to that produced by the second block of rewrite rules. The two-level formalism had advantages in simplifying rules. However, there were difficulties due to the need to force the rules to operate in a deterministic fashion. In a practical text-to-speech system, the rewrite rule formalism would be favoured, despite the greater number of rules and their greater clumsiness, since the critical ordering of rewrite rules easily introduces the necessary determinism.

1 Introduction

In developing a text-to-speech (TTS) synthesiser for the Welsh language, it is necessary to include a set of letter-to-sound rules, to convert words from an orthographic representation to a phonemic transcription. Letter-to-sound (LTS) rules have been part of TTS systems for English for many years (e.g. Allen *et al.* 1987, McAllister 1988), including those systems that carry out a morphological analysis of the input words. This paper describes a set of LTS rules that have been written for Welsh. This appears to be the first attempt at writing such rules for Welsh.

In addition, this paper compares two types of rule formalism for expressing LTS rules. Context-sensitive rewrite rules have long been used for the expression of linguistic rules. This rule formalism is well suited to the task of rewriting an input string into an output string, possibly using a distinct output alphabet. However, it is less suited to the task of grouping an input string into (implicit) units, retaining the same alphabet. Therefore it was decided to experiment with the use of two-level rules for part of the process of letter-to-sound conversion. These are better suited to a grouping task, due to the fact that it is possible to make reference not only to the input form of the context, but also to its output form. PCKIMMO was the software chosen to instantiate two-level LTS rules, a purpose for which it was not directly designed. This appears to be the first attempt at using two-level rules for expressing letter-to-sound rules for any language.

The rules are evaluated over 460 unique Welsh words, and conclusions are then drawn about the relative usefulness of the two formalisms for the purpose of encoding letter-to-sound rules within the context of a text-to-speech synthesis system.

1.1 The role of letter-to-sound rules

In a TTS system for unrestricted input text, the most powerful strategy is to carry out morphological analysis on all input words that can be analysed, and to pass the remainder to a set of LTS rules. This approach has the twin advantages of productivity (in that neologisms based on existing morphemes can be analysed) and comprehensiveness (in that even words not found in the morpheme dictionary can, to some extent, be handled). In the case of English, it is accepted that unaided LTS rules will give rise to many errors, but this is held to be preferable to the lack of any output at all. In the same way, LTS rules for Welsh will also be used for those words where no entries exist in the morpheme dictionary.

In addition, LTS rules for Welsh are of use during the development of a TTS system for Welsh. No machine-readable (or hard-copy) dictionary of Welsh exists that incorporates pronunciation. Therefore it will be necessary to generate phonemic forms for entries by running LTS rules over the orthographic forms of headwords in the dictionary (with subsequent manual checking). It is clear that LTS rules have an important part to play in a TTS system for Welsh.

1.2 Particular features of Welsh

One striking feature of Welsh (at least for English speakers) is the fact that, on the whole, there is a much closer mapping between grapheme and phoneme than is the case in English. This situation could be expected to lead to a higher accuracy for Welsh LTS rules than for English rules, and also a lower number of Welsh rules than English ones.

In the case of consonant graphemes, such is indeed the case, as there is almost a one-to-one mapping between grapheme and phoneme for consonants (even the consonant digraphs are invariant, e.g. `ff' for /f/). In the case of stress assignment, the situation in Welsh is very much simpler than in English: in Welsh polysyllables, the penult is stressed. Of the few exceptions to this rule, the majority are stressed on the final syllable (ultima). The small number of words stressed on the antepenult are largely loan-words from English. In many cases, irregular stress is indicated orthographically by an accent mark. An example is *gwacáu*, input at the keyboard as `gwaca+u', `to empty', /g w a k ai*/, where stress is shown by an asterisk in the phonemic transcription (phonemic symbols are explained in Table I below). Where irregular stress is marked orthographically in this way, it can be handled by the LTS rules. Words with irregular stress that is not orthographically marked will have to be included in the lexicon as exceptions (e.g. *parhau*, `to continue', /p a r h ai*/).

In the case of vowels, Welsh has phonological vowel length, determined by the type and number of the following consonants (Awbery 1984). The contrast between long and short vowels exists only in stressed syllables, as all vowels are phonologically short in unstressed syllables Phonological vowel length is realised phonetically in terms of both vowel quality and duration, in that, except for /a/, phonologically short vowels have lower vowel height and shorter duration (Jones 1984). Phonological vowel length must therefore also must be included in the LTS rules.

One vowel in particular (orthographic 'y'), as well as having long and short forms (/ii/ and /i/) like other monophthongs, may also be realised as schwa (/@/) in non-final syllables of polysyllables. It may also represent the first, vocalic, part of

a diphthong ('yw', /iu/), or the second, consonantal, part of a diphthong ('wy', /ui/). This means that the LTS rules must do a certain amount of (implicit) syllable grouping within each word. In addition, Welsh permits *epenthetic* vowels, these being vowels which are pronounced but which are not shown in the orthography. Such vowels occur in certain phonological contexts: for example, *cefn*, `back', is pronounced in some dialects as /k ee* v e n/ (with stress on the first vowel).

The greatest difficulties, however, are posed by the graphemes 'i' and 'w'. The grapheme 'i' may represent either a vowel (which can be long /ii/ or short /i/) or the palatal glide /j/. The grapheme 'w' may represent either a consonant (/w/), a monophthong (long /uu/ or short /u/), the first half of a diphthong ('wy', /ui/, with vocalic /u/ and consonantal /i/), or a labialisation marker on an alveolar consonant (/l, n, r/) after /g/ or start-of-word (e.g. *gwraig*, `woman', is pronounced with a labialised /r/ as /g rw ai g/). The grapheme 'w' is by far the most difficult grapheme for the LTS rules to handle, and accounts for a great number of the existing rules, due to the many ways in which it can be interpreted.

1.3 Rule software used

1.3.1 Context-sensitive rewrite rules

The rule software used for the first formulation of the LTS rules is publicly available. The main C program of this software is known as `phon.pro': its English instantiation is known as `eng' (there is also a Spanish instantiation). It was originally written by Greg Lee of Hawaii University (e-mail address: lee@uhunix.uhcc.hawaii.edu), who used it to realise a set of LTS rules for English. The software allows the linguist to write critically-ordered context-sensitive rewrite rules in a form familiar to linguists. The software converts these rules into a program header file, and compiles a program (written in the C programming language) which runs the rules. This approach combines the advantages of user-friendliness in writing the rules with speed of running the rules. A few modifications were made to the `phon.pro' program by the present author, in order to allow for Welsh-specific contexts. For example, `D' was introduced to refer to `one and only one consonant grapheme or digraph that can condition phonological length in the preceding monophthong', i.e. one of `b, d, g, f, dd, ff, th, ch'. The rules were developed and run on a Sun 4 workstation, but it is also possible to run them on any platform that has a C compiler.

1.3.2 Two-level rules

The rule software used for the second formulation of the LTS rules is PCKIMMO (originating from the Summer Institute of Linguistics, and available from that source and from various public FTP sites). This is a realisation for PCs of the earlier KIMMO two-level rule software, and is written in C. It is documented in Antworth (1990). The software is also available in a Unix version, and this is the version that was used in the work reported here. In addition, there is a public-domain program, KGEN (also available from the Summer Institute of Linguistics) which converts two-level rules written in context-sensitive form into the state tables used by PCKIMMO. This program exists in versions for PC and Unix, and the Unix version was used in the work reported here. This software was used, rather than the earlier KIMMO, since PCKIMMO is available from several sources without fee. Therefore work carried out using PCKIMMO is likely to have greater comparabillity across a wide range of similar work that will be carried out using this easily-available software.

1.4 Welsh accent used

Welsh exists as a constellation of local accents and dialects, together with a non-local attempted spoken standard form known as *Cymraeg Byw*, `Living Welsh', (mainly used by adult learners), and a more formal literary language that is not used in everyday speech. The many local accents precipitate out into two broad groupings: North Welsh accents and South Welsh accents. North Welsh accents contain a greater number of distinct vowel phonemes than do South Welsh accents, since the former have retained the high central unrounded vowel which in the latter has merged with the high front unrounded vowel. The LTS rules that have been developed for Welsh are based on a South Welsh pronunciation of a dialect that is close to *Cymraeg Byw* (i.e. it has the added feature of epenthetic vowels). It would, however, be a minor matter to edit them so that they output North Welsh transcriptions instead (ithe high central unrounded vowel would easily be differentiated from the high front unrounded vowel, as they are orthographically distinct). More specifically, the rules for insertion of epenthetic vowels would be needed only in some South Welsh accents, such as that from the south central part of Wales near Llandovery. In order to cover a wide range of phenomena, an accent containing epenthetic vowels has been chosen as the target accent.

1.5 Transcription conventions

A Machine-Readable Phonetic Alphabet for Welsh (MRPAW) was designed, along the lines of the Machine-Readable Phonetic Alphabet for English in use at the Centre for Speech Technology Research (see section 6 of Harrington, Laver & Cutting, 1986). In this type of transcription system, each phoneme is represented by one or more alphabetic characters, with a space between each phoneme. As a South Welsh accent had been chosen, there were 32 consonants. These included the three labialised consonants /lw, nw, rw/, and the phonemes /z/ (voiced alveolar fricative), /ch/ (voiceless palato-alveolar affricate), and /jh/ (voiced palato-alveolar affricate) that are used only in loan-words from English. With the 19 vowels, there were a total of 51 phonemes. Table I shows the phonemic inventory and symbols used (assuming a South Welsh accent of Welsh).

(Table I about here)

For the input of orthographic forms, it was decided to assume no special character capability on the part of the computer terminal that might be used for the eventual TTS system. Therefore, acute accent was input as a plus sign (+) after the relevant vowel, grave accent as a backward slash (\) after the relevant vowel, and circumflex (^) and diaeresis (") as the appropriate symbols after the relevant vowel. A plus sign was used for the acute accent rather than a forward slash, as the forward slash is a reserved symbol in PCKIMMO, which was used for the second version of the rules. Some examples follow:

(1) *siòl*, `skull' input as `sio\l', output by LTS rules as /sh o* l/

(2) *siôl*, `shawl' input as `sio^l', output by LTS rules as /sh oo* l/

(3) *sïo*, `to hiss' input as `si"o', output by LTS rules as /s ii* o/

(4) *monópoli*, `monopoly' input as `mono+poli', output by LTS rules as /m o n o* p o l i/

1.6 LTS rule strategy

The LTS rules implemented by Lee for English (using his `phon' software) were based on those in Elovitz, Johnson, McHugh, & Shore (1976), and only one pass through each input word was required. For Welsh, however, three passes are required through each word, with a different set of rules for each pass. The first pass adds any epenthetic vowels to

the input word. This has to be done at the outset so that the phonological syllable count will be faithfully represented in the data received by the second set of rules, which assign stress according to syllable position in the word. The second pass implements rules which locate the stressed syllable and differentiate between the vocalic and consonantal forms of orthographic `w' and `i'. The third pass implements the grapheme-to-phoneme rules proper, which also determine phonological vowel length and differentiate between the two variant pronunciations of orthographic `y'.

2 Rules in rewrite formalism

2.1 Epenthetic vowels

In accents from the south central part of Wales, an epenthetic vowel occurs after one of /b, d, g, p, t, k, f, v, dh, s, x, th/ when followed by one of /l, n, r/, or between /m/ and /l/, at the end of a word (which may form the first element of a compound word). The vowel to be inserted is identical to that of the preceding syllable. If the preceding syllable contains a diphthong, then the epenthetic vowel takes on the quality of the second element of the diphthong. Two examples follow:

(5) *aml*, `frequent', pronounced /a* m a l/

(6) *llwybr*, `footpath', pronounced /lh ui* b i r/

The following is an example of the rules for epenthetic vowels, which comprise the rules implemented during the first pass through the input word. Since the rule formalism does not permit the specification of an output symbol from a null input, it is necessary to supply rules for all possible consonants that can precede an epenthetic vowel, so that the consonant can then form part of the rule's output. Since the quality of the epenthetic vowel depends on that of the preceding vowel, it is also necessary, for each consonant, to specify all possible preceding vowels, both monophthongs and diphthongs. These restrictions lead to the large number of rules required for a relatively simple task, namely 170 rules (including rules for puctuation characters).

(7) e[f]R# = fe

This rule applies to the grapheme `f' when preceded by `e' and followed by one of `l', `n', `r' (represented by the single `R' in the rule), which in turn is followed by a non-alphanumeric character (represented by `#' in the rule), which would signal end-of-word. If the rule applies, the sequence `fe' is output in the place of the input `f'. Thus, for instance, input *cefn*, `back', would be output as *cefen*. It should be noted that, at this stage, the representation of the data is still in orthographic form. The one-letter aliases are glossed in Table 1 below.

In addition to specifying epenthetic vowels, the rules of the first pass also included a block of rules inserting a diaeresis symbol (") between two adjacent non-high vowel graphemes which could not form a diphthong with each other, such as 'eo', 'ao' and 'oa'. These rules also inserted the diaeresis symbol between a diphthong and a monophthong or second diphthong (as in *chwaraewr*, /x w a r ai* u r/, 'player'). Since the diaeresis symbol is used in the orthographic form of some Welsh words, this was a case of generalising an existing phenomenon to all applicable cases. The diaeresis symbol was referred to in some rules of the second pass, when determining the syllables in the context specification.

2.2 Stress and vowel location

The output of the first set of rules forms the input to the second set of rules. These locate the stressed syllable (one per word, except in the case of hyphenated compounds), and also differentiate between the various realisations of orthographic 'w' and 'i'. There are 731 rules altogether (including rules for punctuation characters)

The rules fall naturally into several `blocks' of rules, each dealing with the same kind of context. As the rules are critically ordered, the rules with the most specific contexts appear first, while those with the least (or no) context appear last, and serve as default rules.

2.2.1 General strategy

In the first block of rules, vowels that are orthographically marked as stressed are output as capitalised graphemes (capitalisation being used as a marker of stress in the output forms). Such a syllable may form the ultima, the antepenult, or a preantepenultimate syllable of a polysyllabic word, or may be a monosyllabic word. The stressed vowel may be

orthographically shown by either an acute accent (input as `+') or a circumflex (if the vowel is long). An example rule follows:

(8) [a+u] = AU e.g. *nesáu* = *nesa*+*u*, /n e s ai*/, `to approach'

In the above rule, the capitalisation of the output (which is still in orthographic form) indicates stress. There is no context specification in this rule.

The next block of rules refers to monosyllabic words. The rules assume that all monosyllables are stressed, since stressless monosyllables, forming a limited set of function words, would already be included in the lexicon. The rules allow for vowel contractions, such as word-final "....a'u", pronounced /ai/ (e.g. *dyma'u* from *dyma* + *eu*, `here are their....'). Included here are rules for penult vowels where the ultima vowel is `w' (phonemically /u/), since such a pattern would appear to be a single syllable to the rules.

The third block of rules handles unstressed penults before ultimas that are orthographically marked as stressed, or unstressed preantepenults before stress-marked antepenults. The fourth block of rules, similarly, handles unstressed penults after antepenults that are orthographically marked stressed. The fifth block of rules handles stressed penults (the default case in polysyllables), where no orthographic marking of stress is present. An example rule follows:

(9) [iw]QCVC# = IW e.g. *diwrnod*, /d iu* r n o d/, `day'

This rule states that input `iw' is output as `IW' (i.e. a stressed vowel) in the specified context. Table II glosses all one-character aliases used in the rules, while Table III below glosses some characters used in the output.

(Table II about here)

(Table III about here)

The sixth block of rules handles the remaining vowel cases, i.e. vowels in unstressed ultimas and antepenults (both of which are normally unstressed), and unstressed penults (normally stressed). The seventh block of rules allows consonant graphemes to pass through unchanged, while a final block does the same for punctuation marks.

2.2.2 Orthographic `i'

Orthographic 'i' can be realised as either a vowel or a palatal glide. An example follows:

(10) #C[iai]C# = JAI e.g. *iaith*, /j ai* th/, `language'

The rule in (10) states that the sequence `iai' becomes `J AI' (i.e. a palatal glide followed by the stressed diphthong `ai') in the specified context.

2.2.3 Orthographic `w'

Orthographic `w' is the subject of many of the rules in most blocks of rules. Since `w' cannot be classed a priori as either a consonant or a vowel before the operation of the rules, it is necessary to have separate sub-blocks of rules to deal with the special cases it presents. An example follows:

(11) [w]SGHG# = W e.g. *bwled*, /b u* l e d/, `bullet'

(12) [w]CVC# = W [not actually one of the rules used]

The rule in (11) states that `w' becomes `W' (i.e. a vowel, and identified as stressed by capitalisation) in the specified context: this is the pattern found in *bwled*. The simpler rule in (13), while seemingly adequate to cover the same case, would not in fact do so. This is because it is necessary to specify that at least one consonant must intervene, in order to cut out words such as *dwyn*, /d ui* n/, `to bear', where the `w' forms part of a diphthong, and words such as *chwaer*, /x w ai* r/, `sister', where the `w' functions as a consonant. The rule in (12) is also inadequate because it could not cover cases such as *cwrw*, /k uu* r u/, `beer', where the second vowel is the grapheme `w' which has not yet been assigned consonantal or vocalic status (which is why it cannot with certainty be assigned to the `V').

Complications also arise with the two orthographic sequences `wyw' and `ywy', as in *gwyw*, /g w iu*/, `withered', and *tywyllwch*, /t @ w @* lh u x/, `darkness'. The following are two of the rules for these sequences:

(13) K[wyw]C# = MYW

(14) [ywy]SGwG# = yMY

The rule in (13) states that the sequence `wyw' becomes `MYW' (i.e. consonantal `w' (represented by 'M') followed by the stressed diphthong `yw') in the specified context. This rule would apply to *gwyw* to produce (part of) `gMYW' The rule in (14) states that the sequence `ywy' becomes `yMY' (i.e. unstressed vowel `y' followed by consonantal 'w' ('M') and stressed vowel `y') in the specified context. This rule would apply to *tywyllwch* to produce (part of) `tyMYllwch'.

(15) G[wyw]H = WYM e.g. *dwywaith*, /d ui* w ai th/, `twice'

The rule in (15) states that the sequence `wyw' becomes `WYM' (i.e. a stressed realisation of the diphthong `wy' plus 'M' showing consonantal `w') in the specified context. This rule is ordered after that in (13), and so does not deal with cases with initial `c', `g' or `ngh' (represented by `K' in (13)).

2.2.4 Features of the second set of rules

In the examples quoted above, it can be seen that many rules are implicitly performing a syllable parse of the input, in that they refer to the end of the word plus a certain number of intervening syllables. For each rule where one or more syllables are specified in its left or right contexts, separate versions of the rule must be written for cases with and without orthographic 'w' in each possible location. It is this factor which mainly accounts for the large number of rules (731) needed in the second pass, and which (as seen below) is obviated by the use of two-level rules instead.

2.3 Grapheme-to-phoneme conversion

The output of the second set of rules forms the input to the third (and final) set of rules. These rules number 356 (including the punctuation rules) and are less complex than the second set, though more complex than the first. The rules for consonants are very straightforward, while those for vowels must handle vowel length and the variant realisations of `i', `y' and `w'. The output of the second set of rules distinguishes vowels and consonants where necessary by using the extra symbols `J' and `M' for the consonantal realisations of `i' and `w', and distinguishes stressed vowels by using capitalisation.

2.3.1 Phonological vowel length

In Welsh, phonological length for monophthongs is determined by the type and number of the following consonants (Awbery 1984). Where a vowel is followed by one of orthographic `b, d, g, f, dd, ff, th, ch' (phonemically /b, d, g, v, dh, f, th, x/), or no consonant, before the next vowel (or end-of-word), then the vowel is long if it is stressed (assuming a South Welsh accent). The vowel is also long if followed by one of orthographic `s, ll' (/s, lh/) plus end-of-word (*ibid*.). Where the vowel is followed by any other consonant, or by two or more consonants, then it is short (even if stressed). The following are examples of vowel length rules:

(16)	C[E]Dwy = ee*	e.g. *dedwydd*, /d ee* d ui dh/, `happy'
(17)	C[E]DwV = e*	e.g. *edwi*, /e* d w i/, `to fade'
(18)	C[E]Dw = ee*	e.g. *bedw*, /b ee* d u/, `birch', *credwr*, /k r ee* d u r/, `believer'
(19)	C[E]DiV = e*	e.g. *brechiad*, /b r e* x j a d/, `vaccination'
(20)	C[E]DV = ee*	e.g. *brechu*, /b r ee* x i/, `to vaccinate'
(21)	C[E]V = ee*	e.g. *lleol*, /lh ee* o l/, `local'
(22)	C[E]D# = ee*	e.g. *llech*, /lh ee* x/, `slate'
(23)	C[E]P# = ee*	e.g. *lles*, /lh ee* s/, `benefit'
(24)	C[E]# = ee*	e.g. *lle*, /lh ee*/, `place'
(25)	[E]C = e*	e.g. *llen*, /lh e* n/, `sheet'
(26)	[e] = e	e.g. *llefaru*, /lh e v a* r i/, `to speak'

These (critically-ordered) rules use `D' to refer to the vowel lengthening consonants given above, and `P' to refer to the additional consonants `s' and `ll' (/s/ and /lh/) which lengthen a preceding monophthong only when word-final, in South Welsh accents (Awbery 1984). In (16), (18), (20), (21), (22) and (23), the single vowel-lengthening consonant is followed by a vowel or end-of-word, and so the preceding vowel (which is stressed) is long. In (17), (20) and (25), the vowel-lengthening consonant is followed by a second consonant (which may be a glide) and so the preceding vowel (even though stressed) is short. In (24), the stressed vowel is followed by end-of-word, and so is long, while in (26), the input form of the vowel shows it to be unstressed, and so it is short whatever the context.

2.3.2 Orthographic 'i'

The second set of rules has distinguished between the vocalic and consonantal realisations of the grapheme `i' by outputting 'I' or 'i' for the former and 'J' for the latter. The third set of rules then converts this output to the appropriate phonemes: /i/ or /ii/ (stressed or unstressed) for the vocalic form, and /j/ for the consonantal form. An exception occurs where `s' precedes, in which case the `si' digraph takes precedence over the rules for 'i', and the output is /sh/ (voiceless palato-alveolar fricative).

2.3.3 Orthographic 'y'

Where it is a monophthong, orthographic 'y' becomes schwa (stressed or unstressed) when in non-final syllables of polysyllables, and /i/ or /ii/ (stressed or unstressed) in monosyllables or final syllables of polysyllables. Where it forms part of a diphthong (orthographic `oy, wy, yw, ey'), orthographic `y' is a high front vocoid (functioning in either a vocalic or consonantal manner) and never schwa.

2.3.4 Orthographic 'w'

A final complication in the treatment of `w' is the case of labialised consonants, in words such as *gwlad*, /g lw aa* d/, `country'. In such cases, there is strong labialisation on the consonant following the /g/ (which may be /l/, /n/ or /r/). A labialised consonant may also follow word-initial /ng/ (voiced velar nasal) or be word-initial itself. The latter two contexts are the result of two different types of consonantal mutation (nasal mutation and soft mutation respectively) applying to the root form with initial /g/. One of the rules dealing with labialised consonants is as follows:

(27) #g[Mr] = rw e.g. *gwraig*, /g rw ai* g/, `woman'

Since the second set of rules here outputs the `M' character, the third set of rules is provided in the input string with the information that this is consonantal 'w', and so the rule is straightforward.

2.4 Evaluation of output

The next step was to evaluate the output of the rules over a representative sample of Welsh words. This test was identical for both the rewrite rules and (later) the two-level rules.

2.4.1 Data

The data used for evaluation was taken from an article from a Welsh women's magazine. This was chosen as being representative of the level of language used by Welsh speakers in everyday life. The number of unique words was 460. Inflected or derived forms of the same lexeme were included, as these often introduced a substantial difference into the pronunciation. English words that occurred in the text were excluded. A minimal amount of pre-processing was done, in that any initial capital letters were reduced to lower-case. The rules were run over this data in three passes, as described above.

2.4.2 Results

An initial count of the results showed that 57 of the 460 words were incorrectly output, i.e. a success rate of 88%. However, closer examination revealed various categories of error, as follows.

There were 40 monosyllabic function words among the 57 errors. These words had been assigned a stress by the rules, even though in the vast majority of cases they would not be stressed in actual use. For most of these 40 words, the fact that stress had been assigned was the only error.

There were 14 words among the 55 that were exceptions in Welsh (e.g. *ymddangos*, /@ m dh a* ng g o s/, 'to appear', was output by the rules as /@ m dh a* ng o s/, without the /g/. The appearance of /g/ in this word is a lexical exception.

There were 3 words among the 57 that would have required morphological knowledge for correct specification of pronunciation. For example, *grynwraig*, /g r @* n rw ai g/, 'Quaker woman' was output by the rules as */g r @ n u* r ai g/, where the 'w' grapheme has incorrectly been taken to be a vowel. In fact, it is a labialisation marker, as a morphological boundary intervenes between 'n' and 'w' (the constituent morphemes being the mutated forms of *cryn*, from the verb 'to quake', and *gwraig*, 'woman', with labialised /rw/).

2.4.3 Discussion

Any set of letter-to-sound rules for any language will encounter the second and third categories of error above, as LTS rules cannot handle exceptions and have no access to morphological knowledge. However, the limited set of function words in a language would never be passed to the LTS rules in the first place. This is because these words would be included in the lexicon in any TTS system. Therefore, a fairer test of the functioning of these LTS rules as part of a TTS

system would entail excluding the function words from the analysis. Given this revised interpretation of the data, the results show only 17 errors (the exception words and polymorphemic words) out of 420 input words, i.e. a 96% success rate. This rate is slightly better than the 93% achieved with LTS rules for English (McAllister 1988), and probably represents a realistic upper limit on accuracy for LTS rules.

3 Rules in two-level formalism

3.1 Motivation

In section 2.2.4 above, it is observed that what the rules are doing is an implicit parse of the input segment string into syllable-sized units, with separate versions of each rule for many permutations of presence or absence of 'w' at appropriate locations in the context. Context-sensitive rewrite rules can handle this type of process only somewhat clumsily. This is because they can refer only to the input string and not to the output of any rule. Therefore it is necessary for them to repeat the differentiation contexts for 'w' in each rule that requires it. It is true that the increase in the number of rules is compensated for, in this case, by the speed of execution of the compiled rules in `C' form. However, an alternative strategy would be to undertake syllable parsing using two-level rules, which are able to refer to the output string as well as to the input string. This is the type of approach adopted in several algorithms for morphological decomposition, such as Ritchie, Pulman, Black & Russell (1987), where the lexicon consists of permissible morphemes, and the two-level rules encode `graphotactic' spelling changes at morpheme boundaries (e.g. for English, the occurrence of `ie' for underlying `y' before the inflectional suffix `s' in *tries*).

While such two-level rules have been used extensively in the process of morphological parsing, they do not seem to have been used for phonological rewriting tasks of the type discussed here. Therefore it was decided to experiment with the use of two-level rules as a formalism for expressing letter-to-sound rules, in order to evaluate their effectiveness compared with the conventional `one-level' rules described above. There are two aspects to such an evaluation: formal and practical. An evaluation of the formal aspects of the rule formalism would examine the number of the rules, the power of the rule notation, and the ability of the rules to account for a greater or lesser amount of data than rules in another formalism. An evaluation of the practical aspects of a rule formalism would examine ease of rule-writing, development

time, ease of rule compilation, and the computing resources used in execution of the rules. It is hoped to touch briefly on all these considerations in what follows.

3.2 Revised form of the rules

The second set of context-sensitive rewrite rules were re-written in two-level form, where each element of a rule comprised a correponding pair of a lexical character and a surface character. This entailed extensive recasting of the rules, and there was no simple correspondence between the two versions. Only the second-pass rules were rewritten, as the first and third pass rules had no need to refer to the output string. There were 433 rules in all in the newly-written set, as against the 731 rules of the rule set for the second pass under the previous formalism. In each pair of symbols, the first symbol (before the colon) referred to the input string (the orthographic form plus any epenthetic vowels), while the second symbol (after the colon) referred to the output string (which might contain 'M' or 'J', or capitalised vowels). The rules were run in PCKIMMO's 'generate' mode, where only the rules are used and no lexicon is required.

Nearly all of the two-level rules took the form of `surface coercion' rules, i.e. with a leftward-pointing arrow (Antworth 1990). The implication of this form of rule is that the specified correspondence between underlying and surface symbols must always occur, given the context (i.e. there was no other surface possibility for the underlying symbol in that context), but that this was not the only context in which that correspondence could occur. The `context restriction' type of rule, with a rightward-pointing arrow, would have implied that other mappings of the underlying character were possible, but that this was the only context in which this particular mapping occurred: no rules of this type were used. The surface coercion form of rule was necessary in nearly all cases, as the number of contexts would have made for a very complicated rule if all had been combined into one rule. The rule notation used had the potential power to combine many such rules into one rule (with a double-headed arrow), but there were practical limitations introduced by the fact that the KGEN state table compiler was being used, which had an upper limit of 63 states. Therefore it was often necessary to use several separate rules. Some examples follow: table IV glosses the symbols used.

(28) {a,e,i,o,u}:{A,E,I,O,U} <= [# | -] @:Co* _ @:Co* @:Vos @:Vos* @:Co* [# | ' | -]

(29) {a,e,i,o,u}:{A,E,I,O,U}<= [# | -] @:Co* @:Vos @:Vos* @:Co* _ @:Co* @:Vos @:Vos* @:Co* [# | ' | -]

(30) {a,e,i,o,u}:{A,E,I,O,U} <= @:Vos @:Co @:Co* @:Vos @:Vos* @:Co* _ @:Co @:Co* @:Vos @:Vos* @:Co* [# | ' | -]

(31) {a,e,i,o,u}{A,E,I,O,U} <= _ @:Co* @:Vos @:Vos* @:Co* [# | ' | -] (incorrect rule)

Rule (28) permits the capitalising of one of a, e, i, o or u when this lies in the penultimate syllable of a disyllabic word. Rule (29) carries out the same function when the vowel lies in the penult of a trisyllable, while rule (30) handles the case where the vowel lies in the penult of a word of four or more syllables (it is rare for a Welsh word to have more than four syllables). It may be argued that rule (31) would be sufficient to handle all these cases, and that it is not necessary to specify each case separately. However, it was found that separate specification is, unfortunately, necessary, due to the occurrence of words where the same vowel grapheme appears in two (or more) consecutive syllables. For example, *gwahanol*, /g w a h a* n o l/, 'different', contains the grapheme 'a' in its first two syllables. If a rule such as (31) were used (converted into finite state table form), the first occurrence of 'a' would cause the rule to be in a non-initial state by the time the second 'a' (the one of interest) had been encountered, so that the rule would fail to fire. For this reason it was often necessary to write a rule in more than one format according to different syllable number possibilities, thus adding to the number of rules and the development time.

(Table IV about here)

It will be noted that consonantal 'w' (symbolised by 'M' on the lexical level) is included in the subset of consonants (symbolised by 'Co' in the rules). This means that it is no longer necessary to specify different rules for cases with and without the grapheme 'w' in either its consonantal or vocalic function. A few rules are used to specify the contexts where surface 'w' corresponds to lexical consonantal 'M', and all other rules, using only lexical 'Co' in the context specification, have no need to re-state the appropriate contexts and constraints. This is a real advantage of the two-level formalism as compared to the `one-level' rewrite formalism previously employed. Similarly, it is possible to refer to a capitalised (i.e. stressed) vowel grapheme without the need constantly to re-state the context that led to its capitalisation, due to the ability of a two-level rule to refer to the output string. This is seen in rule (32) below.

(32) {e,y}::I => Vs:Vob ([^:@ | +:@]) _

Rule (32) enables the second grapheme of stressed diphthongs ending in orthographic `e' or `y' to be mapped to a lexical `I' when they immediately follow a lexically capitalised vowel grapheme. The capitalisation of this vowel grapheme will have been specified by some other rule, and so rule (32) has no need to specify more than this minimal context.

3.3 Rule compilation and execution

The rules were compiled into finite state tables using the Unix version of KGEN, a publicly-available package that converts two-level rules into finite state tables. Some hand-editing of some tables was then carried out, in order to optimise their performance. The compiled state tables were read into PCKIMMO (running in its Unix version on a Sun4 workstation). The `t' option of PCKIMMO was used, enabling it to batch-process data from a prepared file and write the results to another file. In the input file, each word (on a separate line) was preceded by the 'g' command (for 'generate'), in order to run the rules over the input word. The input file used was the output of the first-pass rules, which had likewise formed the input to the second-pass rules described in section 2.2: thus the input to both versions of the second-pass rules was identical.

3.4 Results

The output from the two-level rules running under PCKIMMO was identical to that obtained when using rewrite rules for this task. Thus running the third-pass rules also gave identical output. In this sense the two versions of the second-pass rules are completely equivalent.

4 Discussion

4.1 Formal considerations

The formal considerations discussed below are unrelated to the practical details of running the rules in a given text-to-speech system, but will nevertheless have an indirect effect on these practical details.

4.1.1 Number of rules needed

The two-level formalism required far fewer rules than the one-level formalism: 433 rules as against 731. This is mainly due to the ability of two-level rules to refer to both the lexical and the surface levels in a context specification. However, had it been possible to use the full power of the rule notation, there would have been even fewer two-level rules. The use of the full power of the notation was ruled out by the limit on the number of states that KGEN could produce. Therefore, it would be potentially possible to use far fewer two-level rules than were used here.

4.1.2 Power of the rule notation

It is clear that the two-level formalism is a more powerful rule notation, in that reference can be made to both the lexical and the surface levels. In addition, the particular one-level rule notation used shows disadvantages, such as the fact that only a single (capitalised) character can be used to refer to a subset of characters, and the fact that insertion of a new character (i.e. a zero target) is not straightforward. These limitations, however, are not inherent to the one-level formalism, and a more sophisticated one-level program would doubtless be able to overcome them.

4.1.3 Data accounted for by the rule formalisms

As reported in section 3.4, the two rule formalisms can be made to operate in an equivalent way, in that from the same input they produce the same output. In this particular sense neither is to be preferred over the other.

4.2 Practical considerations

In the design of a text-to-speech system, formal considerations are by no means the only kind of consideration to affect choice of software. Below are brief notes on how the two formalisms compared in terms of more practical considerations.

4.2.1 Ease of rule-writing

From the point of view of the human rule developer, the two-level rules were slightly more complicated to write (even though fewer in number). This is because attention had to be paid to both the lexical and the surface levels in the context

specifications. It could also be due to relative unfamiliarity with this formalism as compared to one-level rewrite rules. In any event, the difference was small, so this is not a major consideration.

4.2.2 Development time

In taking into account total development time for the rules (i.e. debugging as well as the initial drafting), it could be said that the one-level rules took longer. This was mainly because of the lack of traceback facilities in the `C' software used (which was designed as a simple rule compiler, and was never intended as a debugging interface). In running KGEN to compile the two-level rules, it was often possible to inspect the output state tables at the point where KGEN failed, and thus to deduce which rule was causing problems. In addition, when running PCKIMMO, it was possible to use the traceback and logging facilities to construct a record of the operation of the rules for a particular input: Inspection of this record usually made it immediately clear which rule was causing run-time problems. These aids to rule development significantly speeded up the development process.

4.2.3 Ease of rule compilation

In terms of the ease of compilation of the rules into their executable form, the two forms of the rules are equivalent. This is because both are available in both PC and Unix versions.

4.2.4 Resources used in rule execution

As regards the execution of the rules, it appears that the one-level rules were faster. Both types of rule were run on a Sun4 under Unix, but the one-level rules were executed in the form of a straightforward `C' program, while for the two-level rules it was necessary to load PCKIMMO and its user interface before executing the rules. In addition, merely loading the state tables into PCKIMMO took at least half a minute of elapsed time. It is true that the functions making up PCKIMMO are themselves written in `C', and are available to the developer for integration into other software, but this would require a significant amount of programming when compared to the ease of execution of the one-level software. In this respect, therefore, the one-level formalism promised to be more attractive as part of a fully-functioning text-to-speech synthesis system.

4.3 Conclusion

In terms of both formal and practical considerations, neither of the two types of formalism is an outright winner. While the two-level formalism has advantages with respect to number of rules, power of the rule notation, and total development time, the one-level rewrite formalism had advantages with respect to ease of the initial drafting of the rules and speed of execution of the compiled rules.

It may be that the ideal situation would be to adapt the source functions of PCKIMMO in such a way as to include the state tables in a two-level rule engine, such that the combined software would execute as a straightforward `C' program. This question perhaps depends on the purpose for which the letter-to-sound rules are written. If speed of execution is paramount (as in a real-time TTS system) then possibly the one-level software would be preferred, and the penalty paid in development time. If, on the other hand, the purpose is to investigate the properties of rules and discover the most economical statement of processes, then possibly the two-level rules would be preferred, and (assuming no re-programming) the penalty would be paid in speed of execution.

5 References

Allen, J., Hunnicutt, M.S. & Klatt, D. (1987). *From text to speech: The MITalk system*. CUP, Cambridge.

Antworth, E.L. (1990). *PCKIMMO: A Two-level Processor for Morphological Analysis*. Summer Institute of Linguistics, Dallas, Texas..

Awbery, G.M. (1984). Phonotactic Constraints in Welsh. In Ball & Jones 1984.

Ball, M.J. & Jones, G.E. (1984). *Welsh Phonology*. University of Wales Press, Cardiff.

Elovitz, H.S., Johnson, R.W., McHugh, S. & Shore, J.E. (1976). Automatic translation of English text to phonetics by means of letter-to-sound rules. US Naval Research Laboratory Report 7948.

Harrington, J., Laver, J. & Cutting, D. (1986). Word-structure reduction rules in automatic, continuous speech recognition. In: *Proceedings of the Institute of Acoustics*, 8, part 7: 451-459.

Jones, G.E. (1984). The Distinctive Vowels and Consonants of Welsh. In Ball & Jones 1984.

McAllister, J.M. (1988). CSTR Text-to-Phoneme Project Status Report. CSTR Internal Report.

Ritchie, G.D., Pulman, S.G., Black, A.W. & Russell, G.J. (1987). A Computational Framework for Lexical Description. *Computational Linguistics*, 13, 290-307.

Acknowledgements

The author gratefully acknowledges the support of a BP Research Fellowship, awarded by the Royal Society of Edinburgh and funded by British Petroleum.

Table I

p	voiceless labial stop
t	voiceless alveolar stop
k	voiceless velar stop
b	voiced labial stop
d	voiced alveolar stop
g	voiced velar stop
f	voiceless labio-dental fricative
th	voiceless dental fricative
h	voiceless glottal fricative
x	voiceless uvular fricative
v	voiced labio-dental fricative
dh	voiced dental fricative
s	voiceless alveolar fricative
z	voiced alveolar fricative
sh	voiceless palato-alveolar fricative
ch	voiceless palato-alveolar affricate
jh	voiced palato-alveolar affricate
lh	voiceless alveolar lateral fricative
rh	voiceless alveolar trill
l	voiced alveolar lateral
r	voiced alveolar trill
w	voiced labial-velar approximant
j	voiced palatal approximant
m	voiced labial nasal
n	voiced alveolar nasal
ng	voiced velar nasal
mh	voiceless labial nasal
nh	voiceless alveolar nasal
ngh	voiceless velar nasal
i	(lax) close front unrounded vowel
e	(lax) half-open front unrounded vowel
a	(lax) open front unrounded vowel
o	(lax) half-open back rounded vowel
u	(lax) close back rounded vowel
@	mid central vowel (schwa)
ii	(tense) close front unrounded vowel
ee	(tense) half-close front unrounded
aa	(tense) open front unrounded vowel
oo	(tense) half-close back rounded vowel
uu	(tense) close back rounded vowel
oi	diphthong: first element vocalic
ai	diphthong: first element vocalic
ui	diphthong: first element vocalic
iu	diphthong: first element vocalic
eu	diphthong: first element vocalic
au	diphthong: first element vocalic
@i	diphthong: first element vocalic
@u	diphthong: first element vocalic

Table I: MRPAW symbols and inventory of phonemes used (South Welsh accent)

Table II

Q	1 consonant (includes w)
S	1 non-w consonant
C	0 or more cons's (includes w)
G	0 or more non-w consonants
V	1 vowel (includes w)
H	1 non-w vowel
R	1 of {l, n, r}
K	1 of {c, g, ngh}
#	1 non-alphabetic character (end-of-word)
D	1 of {b,d,g,f,dd,ff,th,ch}
P	1 of [s, ll}

Table II: One-character aliases used in context specification in rewrite rules.

Table III

J	palatal glide
M	consonantal 'w'
W	stressed vocalic 'w'
w	unstressed vocalic 'w'

Table III: Some characters used in the output of the rules.

Table IV

Co	all consonant graphemes (including 'M')
Vs	upper- and lower-case vowels
Vos	lower-case vowel graphemes
Vob	upper-case vowel graphemes
Vny	lower-case vowels except `y'
VNY	upper-case vowels except `Y'
M	consonantal 'w'
Accf	backslash and circumflex
*	zero or more of
@	any symbol specified by another rule
_	location of the target pair in the string
(x)	optional x
[x | y]	disjunction of x or y
{X, Y}:{x, y}	X,Y correspond with x,y respectively
+	acute orthographic accent
#	boundary character

Table IV: Symbols and subsets used in two-level rules.