Proceedings of The Institute of Acoustics

PHONGRAM: AN INTERPRETER FOR PHONOLOGICAL RULES

Doug Cutting and Jonathan Harrington

The Centre for Speech Technology Research, University of Edinburgh, Scotland.

1. INTRODUCTION

This paper describes a software package, *Phongram*, that enables a phonological grammar to be set up that maps *underlying, phonemic forms* of an input lexicon onto *surface, phonemic forms*. The software is written in Interlisp-D which runs on Xerox 1100 series machines. Phongram takes as its input an *input lexicon* which contains any number of orthographic forms keyed to underlying phonemic forms; the *phonemes* of the language in question which may (optionally) each be defined in terms of a set of features; and any number of *phonological rules* that map the underlying forms onto surface forms (see Figure 1).

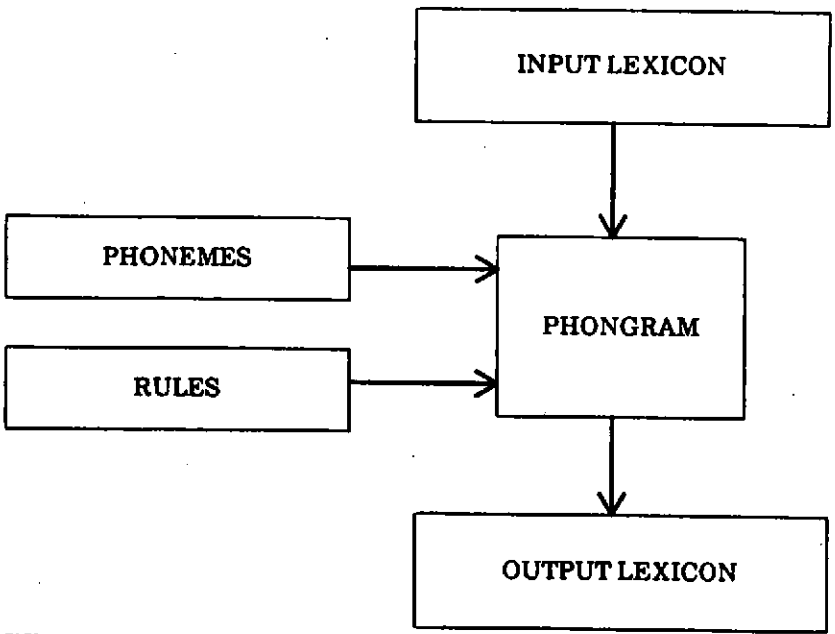

FIGURE 1: RELATIONSHIP BETWEEN INPUT AND OUTPUT LEXICON IN PHONGRAM.

The ouput consists of an *output lexicon* with the original orthographic forms keyed to both underlying and surface forms. The separate stages of Phongram are outlined below; in the final section, some of the applications of Phongram for speech technology and linguistic research are considered.

2. THE INPUT LEXICON

(1) and (2) below are examples of some entries from the input lexicon implemented in [1] for deriving surface forms characteristic of fast, connected speech:

(1) **ability** # @ . * b i . l i . t i #

(2) **punctuation** # p uh ng k . ~ t y uu . * ei . sh @ n #

In this lexicon, underlying forms are made up of the phonemes of Received Pronunciation (R.P.) and *boundary symbols*. There are four boundary symbols: '#' (hash) which denotes a word boundary; '.' (full-stop) which represents a syllable boundary (assigned on the basis of the 'maximum onset principle' - see [2]); and two stress symbols, '~' (tilde) for secondary stress and '*' (asterisk) for primary stress. Phongram accepts many other types of underlying representation; the two main conditions are that a set of phonemes and boundary symbols have been predefined. A possible underlying form, taken from [3], which could be accepted by Phongram is shown in (3) below:

(3) **condensation** # ^{3}k o N = ^{4}d e N s ^{1}A t + i V n #

The superscripts in (3) denote level of stress, and the symbols = and + represent internal and formative boundaries respectively. A task currently in progress is to enable the phonological rules to refer to phonemes that have grouped by internal bracketing. In this case, underlying forms could be encoded using an internal bracketing for relative prominence as in [4][1]:

(4) **Pamela** $[[_{S1}[_{S2}\,p\,a\,{}_{S2}]\,[_{W2}\,m\,@\,{}_{W2}]\,{}_{S1}][[_{W1}\,l\,a\,{}_{W1}]]$

in which S1, S2 are strong nodes and W1, W2 are weak nodes.

INTERPRETER FOR PHONOLOGICAL RULES

2. THE PHONEMES AND FEATURES

The need for phonological rules to refer to features is well established [3], [5]. Phongram enables any set of phonemes to be used as input; and these phonemes may be further decomposed into a *feature matrix*, of the user's choice. Each feature matrix is composed of separate *features*. Each feature, in turn, is a member of a *feature class*. A hypothetical example of the relationship between phonemes, feature matrices and feature classes is shown in Figure 2 below[2]:

CLASS:		CV	CONT	MANNER	PLACE	VOICE	ROUND
/p/	CONS	-CONT	STOP	ANT	-VOICE	-R	
/n/	CONS	SON	NAS	COR	+VOICE	-R	
. . .							

Figure 2: Each phoneme is defined by a feature matrix. A feature matrix consists (in this case) of six features, one feature from each of the feature classes CV (=consonant/vowel), CONT (=continuant), MANNER, PLACE, VOICE, ROUND. Other abbreviations: CONS = consonant; -CONT = non-continuant; SON = sonorant: NAS = nasal: ANT = anterior; COR = coronal; R = round.

It is not necessary for the features to be binary as in [3] and [5].

The rules, as in [3], which may refer to any of the features that have been defined by the user, consist of a *pattern*, or *structural description*, and *action*, or *structural change*, and an optional context in which the rule applies. An example of the rule format is shown in (5) and (6); in all cases, the structural description precedes '= >', which denotes 'rewrites as'; the structural change follows '= >' and precedes '/', which denotes 'in the context of'; the context in which the rule applies always follows '/':

(5)	a	= >	b	/	c	...	d
(6)	(STOP)	= >	(-V)	/		...	(CONS -V)

(5) will replace *a* with *b* when *a* is preceded by *c* and followed by *d* : i.e. *cad* will be replaced by *cbd*. A literal interpretation of (6) is: find a phoneme whose feature matrix contains the feature (STOP) and change its VOICE feature class (see Figure 2 above) to (-V) when the matrix immediately precedes a phoneme whose matrix

includes the features (CONS) and (-V) - i.e. stops become voiceless when they precede voiceless consonants.

Several rules may be conflated[3] into a small number of rules by using optionality and conjunctivity. In (7) below, '< >' brackets denote optionality and '{ }' denote conjunctivity with a comma separating the conjunctive elements[3]. The fully expanded version of (7) is shown in (8a-d):

(7)	a	=>	b	/	c	<d>	---	{e,f}
(8a)	a	=>	b	/	cd		---	e
(8b)	a	=>	b	/	cd		---	f
(8c)	a	=>	b	/	c		---	e
(8d)	a	=>	b	/	c		---	f

'< >' may be nested inside '{ }' to any level, and *vice-versa*. The rules also enable entire phonemes to be deleted or inserted and there is a special feature (NIL) which acts as a 'wild-card' to denote any phoneme the user has defined.

Currently, the software is being developed to include *feature class variables* (which corresponds to *alpha notation* in [3]). This would enable a rule to be written which inserted a stop after a nasal and which made its PLACE feature class (see Figure 2 above) identical to that of the nasal[4]:

(9) 0 => (STOP A.PLACE) / (NAS A.PLACE) ---

Such a rule could be used to insert /p/ after /m/ in a possible production of *dreamt* to give /d r e m p t/ and (in the same rule) /k/ after /ng/ in *length* to give /l e ng k th/.

The software enables selected rules to be run and for rules to be added, deleted or edited on-line. Furthermore, when a rule is run there is an option available to display surface forms as they are generated.

3. THE OUTPUT LEXICON

All phonological rules are numbered, and the numbering specifies the order in which the rules apply. A given rule will apply on both the underlying form and any number of surface forms that have been generated; furthermore, if a given rule r_n generates a surface form SF_1 from a given underlying form and if SF_1 provides a context in which r_n can apply, then r_n will apply. In [1], the single rule 'schwa-deletion' (before sonorant consonants) generates, therefore, 3 surface forms, (see Figure 3) from the underlying form of *additional*:

Underlying form:	/# @ . * d i . sh @ . n @ l #/	
SF1:	/# @ . * d i . sh . n @ l #/	(by schwa-deletion rule applying on underlying form)
SF2:	/# @ . * d i . sh @ . n l #/	(by schwa-deletion rule applying on SF1)
SF3:	/# @ . * d i . sh . n l #/	(by schwa-deletion rule applying on SF2)

FIGURE3: the single rule schwa-deletion derived three surface forms from the underlying form of additional.

The software enables the relationship betwen the underlying form of a given lexical item, all generated surface forms and the rules that have generated those surface forms to be displayed in a tree as shown in Figure 4 below:

INTERPRETER FOR PHONOLOGICAL RULES

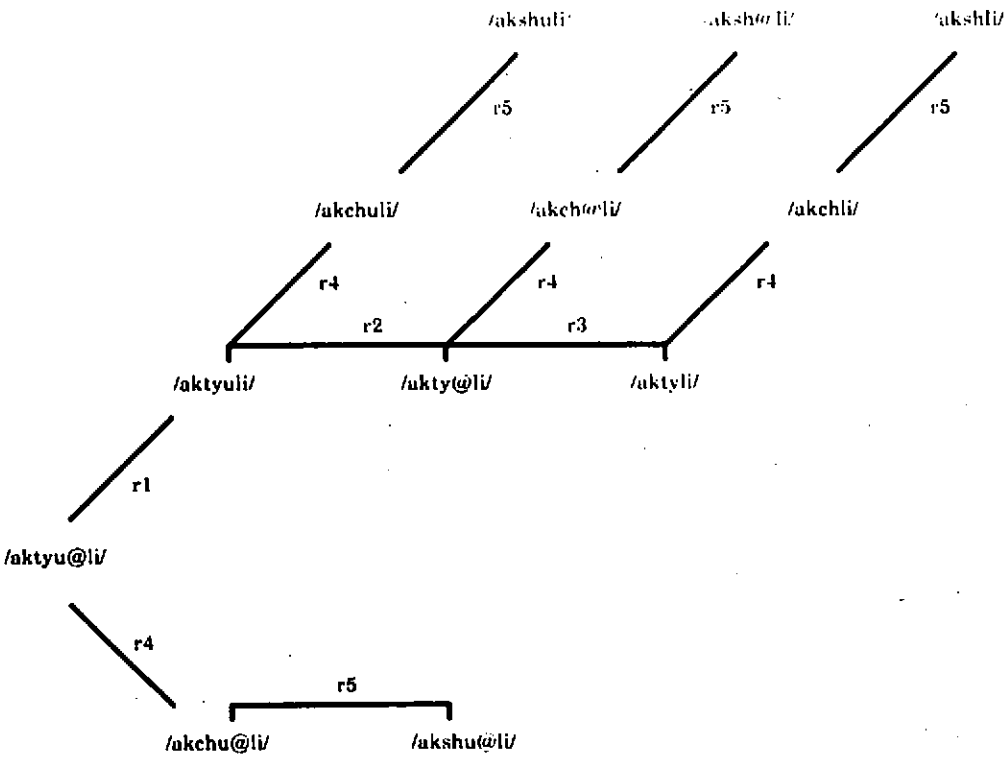

FIGURE 4: The software enables the relationship between the underlying form (in this case /aktyu@li/ - *actually*), the rules (numbered r1, r2...r5) and surface forms to be expressed in a tree-structure.

INTERPRETER FOR PHONOLOGICAL RULES

When all the rules have run, the surface forms are saved together with the original orthographic entry and underlying forms in the output lexicon; all surface forms for a given lexical item which are phonemically identical are automatically removed. In the output lexicon, the underlying form is the first entry, as shown in Figure 5:

```
(accepted
  (#ak.*sep.tid#)          =      underlying form
  (#uk.*sep.t@d#)
  (#@k.*sep.tid#)
  (#@k.*sep.t@d#))

(accident
  (#*ak.si.d@nt#)                 =      underlying form
  (#*ak.si.d@n#)
  (#*ak.si.dnt#)
  (#*ak.si.dn#)
  (#*ak.s@.d@nt#)
  (#*ak.s@.d@n#)
  (#*ak.s@.dnt#)
  (#*ak.s@.dn#))
```

FIGURE 5: a fragment of an ouput lexicon containing orthographic, underlying and surface forms.

4. CONCLUSIONS

The principle use of Phongram at Edinburgh University has been for the generation of reduced and assimilated forms in an automatic speech recognition system. But the generation of reduced and assimilated forms would also be an asset in a phoneme-based, text-to-speech system: if such forms were included, synthetic speech could be generated that corresponded to speech produced at different tempos. Furthermore, although all the rules discussed in this paper and in [1] generate surface forms which are *phonemically* encoded, there is no reason why Phongram could not be used for the generation of *phonetic* strings from a lexicon that contained phonemic, underlying forms. Finally, one of the main advantages of Phongram is that the lexicon, phonemes, features and rules are defined by the user. This means that Phongram can be used to derive surface forms in any natural language; furthermore, the possibility to implement frameworks such as that proposed in [3] makes Phongram suitable for research and teaching purposes in Theoretical Linguistics.

5. REFERENCES

[1] HARRINGTON J.M., LAVER J. & CUTTING D. (1986) Word-structure reduction rules in automatic, continuous speech recognition. *Proceedings of the Institute of Acoustics* (This Volume).

[2] KAHN D. (1976) *Syllable-Based Generalisations in English Phonology*. Indiana University Linguistics Club: Bloomington

[3] CHOMSKY N. & HALLE M. (1968) *The Sound Pattern of English*. Harper & Row: New York

[4] LIBERMAN M. & PRINCE A. (1977) On stress and linguistic rhythm. *Linguistic Inquiry* 8, 249-336.

[5] JAKOBSON R., FANT G. & HALLE M. (1952) *Preliminaries to Speech Analysis*. Technical Report 13, Acoustics Laboratory: MIT.

6. NOTES

1 Liberman & Prince [4] use tree-structures, rather than internal bracketing in defining relative prominence.

2 Chomsky & Halle [3] use long rectangular brackets to define feature matrices. The relationship between their system and the one implemented here is shown in Figure 6 below:

(F1 F2 F3 . . Fn)

FIGURE 6: F1, F2..Fn are an arbitrary number of features. The feature matrix on the left, used in [3], is equivalent to the matrix on the right, used in Phongram.

3 Chomsky & Halle [3] use round brackets '( )' to denote optionality (equivalent to the use of '< >' in Phongram) and as in Phongram '{ }' brackets are used for conjunctivity.

4 As in Chomsky & Halle [3], the structural description is 'zero' for insertion; the literal interpretation of this rule is, therefore: 'zero' is replaced by (STOP)... For deletion, the zero appears in the structural change, the phonemes to be deleted in the structural description.