23

BOLT BERANEK AND NEWMAN INC

CONSULTING · DEVELOPMENT · RESEARCH

AFCRL-68-0085

A PHONOLOGICAL RULE TESTER

Daniel G. Bobrow
J. Bruce Fraser

Contract No. F19628-68-C-0125
Project No. 3668
Task No. 866800
Work Unit No. 86680001

Scientific Report No. 5

This research was sponsored by the Advanced Research Projects Agency under Order No. 627

31 January 1968

Contract Monitor: Hans Zschirnt
Data Sciences Laboratory

Prepared for:

AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
OFFICE OF AEROSPACE RESEARCH
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS

37

CAMBRIDGE NEW YORK CHICAGO LOS ANGELES

AFCRL-68-0085

A PHONOLOGICAL RULE TESTER

Daniel G. Bobrow
J. Bruce Fraser

Bolt Beranek and Newman Inc
50 Moulton Street
Cambridge, Massachusetts 02

Contract No. F19628-68-C-0125
Project No. 8668
Task No. 866800
Work Unit No. 86680001

Scientific Report No. 5

This research was sponsored by the Advanced Research Projects Agency under Order No. 627

31 January 1968

Contract Monitor: Hans Zschirnt
Data Sciences Laboratory

Prepared for:
AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
OFFICE OF AEROSPACE RESEARCH
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS

TABLE OF CONTENTS

ABSTRACT

In this paper, we report on the design and implementation of a system to alleviate the problem of rule evaluation for the linguist in the area of phonology. The system permits the user to define, on-line, sets of rules statable within the framework presented in The Sound Patterns of English (N. Chomsky and M. Halle, 1968), to define phonemes as bundles of specified distinctive features, to define data as strings of phonemes with associated grammatical structure, to test the effect of applying rules to the data, and to store both the definitions and results. The system was written in BBN LISP (Bobrow et al. 1967) on the Scientific Data System 940 computer. The rule application facility described in detail later was implemented by translating the linguistic rules to rules in FLIP (Teitelman, 1967), a format directed list processor embedded in LISP. This made the system construction easy while providing very sophisticated capabilities for the linguist. The system is designed to be used on-line in interactive fashion, with control returned to the user after each command is executed.

BLANK PAGE

INTRODUCTION

Linguistics has as one of its major goals the development of a theory of language which is powerful enough to accurately and precisely characterize the linguistic facts of a natural language. Currently, the most highly developed and potentially most adequate theory of language is that introduced by Chomsky (1957) and involves the concept of a transformational grammar. Although this theory has been designed and modified to enable the linguist to state generalization about a language in a simple and revealing way, an account of some significant portion of a language often results in a complex and interdependent set of rules. Consequently, it becomes more difficult for the linguist to evaluate the work he has done. In fact, linguists have reached the point today where the detail of analysis makes it impracticable to evaluate by hand even a small set of rules.

In this paper, we report on the design and implementation of a system to alleviate the problem of rule evaluation for the linguist in the area of phonology. The system permits the user to define, on-line, sets of rules statable within the framework presented in The Sound Patterns of English (N. Chomsky and M. Halle, 1968), define phonemes as bundles of specified distinctive features, to define data as strings of phonemes with associated grammatical structure, to test the effect of applying rules to the data, and to store both the definitions and results.

The system is written in BBN LISP (Bobrow et al., 1967) on the Scientific Data System 940 computer. The rule application facility, described in detail later, is implemented by translating the linguistic rules into rules in FLIP (Teitelman, 1967), a forma directed list processor embedded in LISP. This makes the system construction easy while providing very sophisticated capabilities for the linguist. The system is designed to be used on-line in an interactive fashion, with control returned to the user after each command is executed.

We point out that this system has been designed to provide a phonological rule testing capability as opposed to a syntactic rule testing system, several of which have developed elsewhere. (See Blair, 1966; Friedman, 1967; Gross, 1967; and Londe and Schoene, 1967) However, because of the modular way in which our system has been designed, it can be made applicable to syntactic rules by extension rather than redesign.

Definitional Capabilities

Within the framework of generative grammar, the role of the phonological component is to interpret the output of the syntactic component and convert this into an appropriate phonetic representation. Thus, the linguist in formulating a phonological rule is concerned first with identifying a relevant phrase marker--a tree structure having phonemes and grammatical markers as the symbols of its terminal string--and then in converting this string into a phonetic representation. For example, a linguist writing a set of rules to account for the assignment of the correct phonetic form for English noun plurals is concerned with relating the following types of strings

"bite"	[bayt]+PL	[bayts]
"tiff"	[tif] +PL	[tifs]
"lid"	[lid]+PL	[lidz]
"love"	[lov]+PL	[ləvz]
"fish"	[fiš]+PL	[fišɨz]
"buzz"	[bəz]+PL	[b zɨz]

As the examples show, there are three plural endings, [s], [z], and [ɨz]. Examination of English noun plurals quickly reveals that plural assignment is not at all random but depends solely on the phonetic form of the last phone in the noun singular form. The following three phonological rules generate the correct plural endings.

$$\text{R1} \quad \text{PL} \rightarrow [\text{ɨz}] \ / \begin{bmatrix}-\text{vocalic}\\+\text{strident}\\-\text{grave}\end{bmatrix} \_\_\_$$

$$\text{R2} \quad \text{PL} \rightarrow [\text{s}] \ / \ [-\text{voice}] \ \_\_\_$$

$$\text{R3} \quad \text{PL} \rightarrow [\text{z}]$$

These three rules are ordered such that the rule R1 is tried first to any noun+PL sequence. R1 states that the plural morpheme, PL, is converted into the phonetic form [ɨz] if it (the PL) follows a phone which is non-vocalic (all consonants except "l" and "r"), is strident (phones with a hissing or hushing sound) and is non-grave (phones which do not have a place of articulation at the front (e.g., "f", "v", "p") or the back ("k", "g") of the mouth. R2 states that PL is converted into the phone [s] after any non-voiced phone. Since R1 has already been tried, it may be the case that R2 cannot apply because the plural marker has already been converted. For example, R1 when tried to the form [fiš]+PL is found to be applicable and the PL is converted into [ɨz]. If this conversion had not occurred, R2, when tried, would have been found applicable and have converted PL into [s], thus causing the unacceptable plural noun [fišs]. R3 is tried after both R1 and R2 and will be applicable in case neither of the first two rules has been applied, since R3 simply converts PL into the phone [z]. A discussion of this set of rules as well as a very insightful alternative can be found in Keyser and Halle, 1968. For a very thorough and detailed analysis of the phonological component of a generative grammar and English phonology, the reader is referred to Chomsky and Halle, 1968.

For a phonological rule system to satisfactorily simulate the effect of a set of phonological rules we need at least three definitional capabilities: for phonemes; for P-markers (or trees, as we shall henceforth refer to them); and for rules. We now examine these definitions in that order.

In keeping with the framework of Chomsky and Halle, 1968, a phoneme is defined as a set of distinctive features. Each distinctive feature (e.g., vocalic, consonantal, strident, fricative) has an associated specification which is normally marked either plus (+) or minus (-), but which in the case of the prosodic features of stress and intonation may take a numerical value such as Ø, 1, 2, 3, ...,

Within our phonetics system, a phoneme is represented by a list of just those features which are marked plus (or have other non-negative value). Thus a phoneme /A/ which was marked positive for the features BACK, LOW, and VOC (vocalic), and a STRESS value of 2 would be represented as the list

```
(VOC BACK LOW STRESS 2)
```

Numerical values of features immediately follow the feature names; all features in the system not included in the list defining the phoneme are assumed marked minus, including STRESS.

Phoneme names in the system can be any string of characters not containing parentheses, brackets, commas or spaces. For the most part, it is possible to use the orthography normally found in the linguistic literature with certain exceptions due to the teletype character set. For example, the phoneme /ʃ/ might be rendered as SH -- all teletype characters are printed in upper case -- /æ/ might be rendered as AE, etc.

To define a phoneme within the system, the user types

```
DPHON <Name> <Phoneme Definition>
```

where <Name> is the phoneme name, and the <Phoneme Definition> is the list of positively specified features. As an example, consider the set of distinctive features CNS (consonantal), FRT (front), HISS (hissing), and the three phoneme definitions:

```
DPHON P ( CNS FRT )
DPHON F ( CNS FRT HISS )
DPHON K ( CNS )
```

These definitions have the effect of the following more familiar linguistic definitions:*

```
/P/ = ( + CNS + FRT - HISS )
/F/ = ( + CNS + FRT + HISS )
/K/ = ( + CNS - FRT - HISS )
```

Finally, we remark that no provision has been made within the system to differentiate between phoneme and phone specification. The user must define phones using the same instruction. This results in a list of phonemes defined to include both those sets of specified distinctive features which are interpreted by the linguist as phonemes, and those interpreted only as phones. There appears to be no difficulty in combining the two types of entities and, as we will see below, it permits a much more efficient output of the steps of a derivation.

* In all examples we refer to segments as having +, -, or numerical specifications although within the system a segment contains only the names of features for which it is non-negatively specified.

Tree Definition

Phonological rules operate on trees rather than phonemes in isolation. These trees are represented in our system by lists; as an example, the tree [bayt]+PL discussed earlier is represented as

```
((N) B AY T + PL)
```

To define a tree in the system, the user types

```
DTREE <Name> <Tree Definition>
```

where the syntax for tree can be represented as

$$\langle\text{Tree}\rangle = ((\langle\text{Syntactic Marker}\rangle^{n})\ \langle\text{Segment}\rangle^{n}) \quad ((\langle\text{Syntactic Marker}\rangle^{n})\ \langle\text{Tree}\rangle^{n})$$

We have used here an extended form of the standard BNF. The superscript n following a name indicates that one or more of the items may occur. A <Segment> is either a phoneme, specified by its phoneme name or its phoneme definition, or a non-phonetic atomic symbol (such as # above) which is used as a marker. The phoneme definition, (the list of positively specified features) is used in place of the phoneme name in the internal representation of a tree.

A tree is a list containing first a syntactic marker, followed by the components which make up this syntactic entity. Often the syntactic marker is composed of more than just a single category such as N (noun). For example, if an English noun were marked for gender, the noun "bite" could be represented as:

```
((N NEUT) B AY T)
```

Note that any information concerning rule exception features would be part of the syntactic marker.

A second type of tree definition is used to create a set of data. Typing

```
     DTREE TØ (ALL T1 T2 . . . TN)
```

defines TØ as the set of trees T1, T2, ... TN. Any rule applied to TØ is applied to all these trees in sequence. This naming schema is very useful when the data remain constant but the rule definitions are being altered.

Rule Definition

A phonological rule identifies a small substring of a phonemic string; if applicable to a given tree, the rule effects some change in this substring, for example, deleting part of it, or adding a phoneme to it. A rule is defined within the system by typing

```
     DRULE <Name> <Rule Definition>
```

The form of rule definitions in our system closely parallels that found in current linguistic literature, both in terminology and notation. Certain differences arise because of the limited character set of the teletype and because of certain assumptions underlying the characterization of a rule. These will become clear in the following discussion.

We distinguish three types of phonological rules within the system: a simple rule, an insertion rule and a string rule. It is convenient to think of each rule as consisting of a left hand side (LHS) which specifies the condition on the substring to be altered, a right handed side (RHS) which specifies the change to be made and a context which specifies the environment in which the substring matched by the LHS must be located.

Simple Rule

A simple rule has the form (<segment><segment><context>)
The LHS of a simple rule specifies a single segment which is identified by either a phoneme name (e.g., A), an undefined symbol name (a non-phonetic symbol, e.g., #), or a bundle of specified features (e.g., (+ VOC - CNS))*. The RHS of a simple rule also specifies a single segment, identified by one of the three forms of the LHS or by the symbol Ø which indicates deletion of the LHS element. A segment of a tree is matched by the LHS of a simple rule under any of the following conditions:

1) if the LHS is a phoneme name and the segment has the same name;
2) if the LHS is a non-phonetic symbol and the segment is this same symbol;
3) or if the LHS is a bundle of specified features and the segment contains corresponding feature specifications for all features specified.

* All feature specifications must be separated from the feature name by a space. A parenthesis has the value of one space.

Every segment of the tree matched by the LHS of a rule is marked, not just the first one encountered. Although in defining a phoneme a feature specification for a phoneme may have only a +, -, or numerical value in a rule, we also permit the values <Name>, (<Name>) and (-<Name>) to occur in a phoneme specification. These named values function as the α, β, γ specification in the literature; that is, as variables whose values are equal or not equal to other variables having the same name. For example, a segment specification (X VOC), matches any phoneme, but the system associates with the name, X, the value of the specifications of the feature VOC in this phoneme. If the specification (X) is encountered later (to the right in the string pattern used in the match), the associated value of X is used in matching the current phoneme. Only if the value of the two specifications are identical does the second segment match the string, assuming all other requirements are met. Thus, the two segments

(X VOC - FRT) (+ CNS (X) FRT)

would match the substring consisting of

(+ VOC - FRT) (+ CNS + FRT)

but not the substring

(+ VOC - FRT) (+ CNS - FRT)

Note that the value of x was picked up from the feature VOC, but used to match with the feature FRT in this example.

The specification (-<Name>) is interpreted similarly, but indicates that the value of the second specification must be different from that of the first.

A marked segment in a tree is changed in the following way:

1) if the RHS of the simple rule is a phoneme name or non-phonetic symbol this item replaces the marked segment in the P-marker
2) if the RHS is a list of phoneme names and/or non-phonetic symbols (but not a bundle of specified features) starting with a colon, e.g., (: A B) all these items are inserted for the LHS
3) if the RHS is a bundle of specified features, the marked segment is changed to reflect that set of specified features
4) if the RHS is Ø the marked segment is deleted

Marking of all identified segments is done first and then all the changes are made.

The following have been constructed to illustrate these cases. (in the format to be typed to the system):

Rule	Comment
DRULE R1 ((+ VOC + VOICE) (- VOICE))	Every segment marked (+ VOC + VOICE) is now marked (- VOICE)
DRULE R2 (O (- VOC))	Every occurrence of phoneme O is marked (- VOC))
DRULE R3 (A E)	Each occurrence of a phoneme segment A is replaced by an E)

DRULE R4 (# (+ SEG))	The nonphonetic symbol # is replaced by a phonetic segment with just feature SEG marked +)
DRULE R5 ((+ VOC) Ø)	Every segment marked (+ VOC) is deleted)
DRULE R6 (E (: A R))	All occurrences of the phoneme E are replaced by the sequences of two phonemes A and R

The simple rules shown above operate on all occurrences of an item in a phoneme string matching the LHS of the rule. However, the user can restrict the domain of this change by specifying a <context> for the LHS for which the rule is applicable. The <context> is stated in the rule definition by following the LHS and RHS by

```
/<Left Context> -- <Right Context>
```

where "--" marks the position of the LHS in this context. Either or both contexts may be empty.

The LHS is inserted in the context for the --. This sequence of <Left Context> -LHS-<Right Context> can be considered a pattern which will match a substring of the phoneme string if, and only if, each individual elementary pattern matches consecutively a segment of the phoneme string. We discuss these elementary patterns below. The implementation of the matching process utilizes this entire string pattern, with only one distinction made for the LHS pattern; a special mark is inserted before LHS pattern to save its matching position in the phoneme string if a complete match is successful. Matching substrings are always found in left to right order. This is important to remember in utilizing "named" feature specifications, as mentioned earlier, and in similar naming conventions discussed later.

The rule R7, defined by typing:

```
DRULE R7 ((+ VOC)(+ STRESS) / (+ VOC - STRESS)(+ CNS) -- (+ CNS))
```

causes any vowel segment to become stressed which immediately follows an unstressed vowel and a consonant and immediately precedes another consonant.

Note that when R7 is applied to a phoneme string of alternating unstressed vowels and consonants, all vowels but the first will be made (+ STRESS) since changes are made only after all substrings matching the string patterns have been found. However, by replacing --, the LHS position mark, by ++, one can specify that the change is to be made in the phoneme string as soon as each match is found. In this case, the result would have only the second, fourth, ..., vowel becoming stressed.

Finally, if a rule has been defined as above, contexts may overlap; that is, the string of segments in the tree identified as part of an acceptable right context for an occurrence of the LHS of the rule may function as the left context for another occurrence of the LHS. To avoid having overlapping of context the user can use // instead of / in the rule definition. In the following rule

```
    DRULE R8 ((+ VOICE) (+ HIGH) // (+ VOICE ) --)
```

only alternating elements of a string of voiced segments will be made (+ HIGH) since the LHS is prevented from acting as a left context by the //.

In addition to the three types of segments which can make up the LHS of a rule, there are a number of other elementary patterns which can be used in the specification of the context or the RHS.

1) (@ $\langle$Synmk$\rangle_1$...$\langle$Synmk$\rangle_n$) This elementary pattern may only be the first element in the context and specifies that this rule is applicable only to a tree (phoneme string) marked with <u>all</u> the syntactic markers $\langle$Synmk$\rangle_1$...$\langle$Synmk$\rangle_n$ (but perhaps others too).

2) $ Equivalent to the variable X in linguistic notation. Will match any number of elements of the substring (including zero, which is tried first).

3) (! <Name><Specfeat>^{n})
(= <Name><Specfeat>^{n})
Used in conjunction with each other. (! <Name><Specfeat>^{n}) matches any segment having the <u>n</u> specific features listed and associates <Name> with this segment. (! <Name>) associates <Name> with any segment. (= <Name><Specfeat>^{n}) matches any segment having exactly the same feature specification as the seg-

ment already associated with <Name>, except for the specified features listed. (=<Name>) matches a segment identical to the one already associated with <Name>. The pattern (! X A STRESS) (= X (-A) STRESS), for example, matches any two phonemes which are identical except for the value of the feature STRESS.

It is important to recognize that because a string pattern is examined by the system in a left to right order, the user, in formulating a rule utilizing this naming convention, must ensure that the associating of the name X, (! X...), occurs in the pattern to the left of a pattern which requires the associated value of X, i.e., (= X...). The first elementary pattern (! X...) calls attention to a segment and associates the name X with it; the second only compares the composition of the segment to the one already associated with the name X, taking into account the differences indicated. If the order of the patterns is reversed, the rule will never apply to any string.

The use of (= <Name><Specfeat>^{n}) in the RHS should be obvious from the preceding discussion. A segment (= X - BACK) in the RHS causes a copy of the segment associated with X to be used on the RHS, with the feature BACK specified negative. The rule R9

((+ VOC)(= X - STRESS) / (! X + VOC + STRESS)--)

replaces a vowel following a stressed vowel by an identical, unstressed copy of that vowel.

4) (EITHER <Segment>$_1$ OR <Segment>$_2$ OR ...)

Used to indicate that the segment matched in the string may be either that specified by <Segment>$_1$ or by <Segment>$_2$, etc. One of the options must be matched. Note that <Segment>$_1$ etc. may contain any number of subpatterns e.g., (EITHER (+ VOC)(- VOC) OR (# (- VOC))).
This disjunctive specification may be used as well to designate possible syntactic markers of a tree. The specification (EITHER (@ N PL) OR (@ ADJ)) immediately following the slash, /,

restricts the application of the rule to trees having either the syntactic markers N and PL, or one having the marker ADJ.

5) (OPT <Segment>)

Used to indicate a possible but not required occurrence of <Segment>. The <Segment> may be a simple segment as described in the discussions of the LHS of a simple rule or may be compound, built up out of the basic patterns we are presently discussing. The alternative with the <Segment> present is tried first. The specification

(+ VOC)(OPT(EITHER (+ VOC) OR (- NAS)))(+ CNS)

matches a segment marked as (+ VOC) followed optionally by a segment marked either as (+ VOC) or (- NAS) all followed by a segment marked (+ CNS).

6) (#<Num><Num><Segment>)

Used to specify a number of successive occurrences of <Segment>. The first <Num> indicates the lower bound, the second the upper bound. If only one <Num> is present, it is interpreted as the lower bound with the upper one indefinite. If no <Num> precedes <Segment> the case (#<Segment>)--the <Segment> need not occur, one segment may occur, two <Segments> may occur .... Note that the pattern (OPT <Segment>) is equivalent to (# 0 7 <Segment>). As an example, the context /(# 1 3 (+ VOC)) -- requires that the LHS occur after at least one but no more than three segments marked (+ VOC).

7) (SIDE <Pred>)

Used to place conditions on features like stress which may have a numerical specification. We allow <Pred> to be either an elementary predicate, or a boolean combination of predicates using (NOT<Pred>), (AND<Pred>$_1$ ... <Pred>$_n$) and (OR<Pred>$_1$ ... <Pred>$_n$). The two elementary predicates available are: (N= <VAL>$_1$ <Val>$_2$ interpreted as "not equal" of the two <Val>s (which must be names used in specifications); and (SLE<Val>$_1$ <Val>$_2$) interpreted as <Val>$_1$ is a Stress Less than or Equal to <Val>$_2$. As an example of a side condition, consider a context

/(A STRESS) (# (+ CNS))(B STRESS)--(SIDE (N= A B))

This context requires that the LHS match a substring only if it follows two vowels of unequal stress which are separated by any number of consonants.

Insertion Rule

A second type of phonological rule is the insertion rule, which is flagged by having a LHS which is just the symbol Ø (zero). This marker, Ø, matches a null segment every place in the string which satisfies the context (before every segment if no context is given). This null segment is the one modified by the RHS. Thus, if the RHS consists of a single phoneme, the effect of the rule is to insert this phoneme in the place specified by the context. As an example:

```
DRULE R10 (Ø (: + A) / (@ N) --#)
```

will cause the sequence + A to be inserted at the end of every tree having the syntactic marker N.

String Rule

In order to allow a rule to change more than one segment in a string, we introduce a string rule. In this rule, the RHS is a list starting with "*" followed by any number of RHS formats. Each succeeding RHS format effects a change on a successive element of the phoneme string (with two exceptions described below). The change starts at the position matched by the LHS element. The LHS may itself specify more than one element as a list starting with "*"; however, all but the first can equally well be put in the right context. For example:

```
DRULE R11 ((*(+ VOC) (+ VOICE)) (* (+ STRESS) Ø))
```

and

```
DRULE R12 ((+ VOC) (*(+ STRESS)Ø) / -- (+ VOICE))
```

have the identical effect of stressing the first and deleting the second of any sequence of segments matching (+ VOC)(+ VOICE).

A special RHS element which may only appear in a string rule is (Ø <Segment>) which inserts the segment specified by <Segment> before the segment in the phoneme string corresponding to this elementary RHS pattern.

The specification *I may be used on the RHS of a string rule to indicate no change to the corresponding segment in the phoneme string. For example, the rule

```
     DRULE R15 ((*A(+ CNS - FRT + STRID) B)(*(- STRESS) *I Ø))
```

marks the A as (- STRESS), leaves the second segment as is, and deletes the third segment, B.

Another segment specification, *M, can be used on a RHS to affect more than one item. The *M stands for metathesis and has the following interpretation. The segment of the P-marker matched by the LHS pattern corresponding to the *M is permuted with the following segment. In order to alter either of these permuted elements, the form (*M <Segment>$_1$ <Segment>$_2$) is used, where <Segment>$_1$ and <Segment>$_2$ correspond to the segments in the permuted order, not the original order in the P-marker. The rule

```
     DRULE R13 ((* A B) *M)
```

permutes all occurrences of A and B; and the rule

```
     DRULE R14 (R (* *I  (*M (+ FRT))))
```

permutes any two elements following an R and marks the one that becomes adjacent to the R (+ FRT). Nothing is done to the segment which originally followed the R.

Sequencing Rules

In the preceding, we have discussed the types of phonological rules and their specifications. Now we introduce a notation for sequencing rules. The command

```
     DRULE RØ (ALL R1 R2 R3 ... RN)
```

has the effect of defining the rule RØ as the list of rules (R1 R2 R3 ... RN) with the following interpretation. When RØ is applied to some tree T1, first the rule R1 is applied to T1 and the result, $T1_{R1}$, is either a new tree (in case R1 was applicable) or T1 itself. Then R2 is applied to $T1_{R1}$, then R3 to the result $T1_{R1\ R2}$, and so on until RN has applied to $T1_{R1\ R2\ ...\ Rn-1}$. Essentially, this (ALL ...) form of a rule definition allows the user to define a set of rules and cause them to be applied in succession.

The instruction

```
     DRULE RØØ (ANY R1 R2 ... RN)
```

has a slightly different interpretation. As before, R1 will apply first to T1, then R2, etc. but the <u>first time</u> some rule is applicable the operations are carried out and the application of RØØ is finished. For example, if RØØ = (ANY R1 R2 R3) and if R2 were applicable, R3 would never be applied to $T1_{R1\ R2}$. Note R1 could not have been applicable or R2 would not have been tried).

Any of the rule names within an (ALL ...) or (ANY ...) form can be one of these forms. Thus the command

```
     DRULE RØØØ (ALL R1 (ANY R2 R3) R4)
```

is a well formed rule.

Testing Capabilities

There are two testing modes in the system: TEST and WTEST. These stand for test and watch test, respectively. Suppose we have

```
R1   = (+ CNS - CONT - VOICE)
       (+ CONT + VOICE)
       / (+ VOC) -- (+ VOC)

T1   = ((N) # P A P A #)
```

where R1 is a rule which makes a voiceless, non-continuent consonant voiced and continuent in intervocalic position. The instruction TEST R1 T1 will cause the system to respond with

```
TESTING
R1
((N) # P A P A #)

((M) # P A P A #)
((N) # P A B A #)
```

where the last line is the result.

* The rules and data in this section are adapted from Rogers, 1967.

If the result of a TEST command contains some segment composed of a bundle of specified features which has no phoneme name, this bundle of specified features will be printed instead of a single equivalent name since none has been defined. It is for this reason that we require that phonemes and phones be defined in the same way and not be distinguished formally within the system.

The command WTEST provides the added feature that the result of each step of the derivation is shown to the user. This is most useful in tracking down exactly where a set of rules is producing unexpected results. For example, suppose we have the following rules:

1. $$\begin{bmatrix}\text{+CONT}\\ \text{-CONT}\\ \text{-VOICE}\end{bmatrix} \longrightarrow \begin{bmatrix}\text{+CONT}\\ \text{+VOICE}\end{bmatrix} \;/\; [\text{+VOC}] \;\text{--}\; [\text{+VOC}]$$

2. $$[\text{+VOC}] \longrightarrow [\text{+STRESS}] \;/\; \left\{\begin{matrix} \# \; [\text{+CONS}] \;\text{--}\; [\text{+CONS}]_1^2 \; [\text{+VOC}] \; \# \\ [\text{+CONS}][\text{+VOC}] \; [\text{+CONS}]_1^2 \;\text{--} \end{matrix}\right\}$$

3. $$X \longrightarrow [\text{-VOICE}] \;/\; \left\{\begin{matrix} \text{--} \quad \# \\ \begin{bmatrix}\text{--}\\ \text{+VOC}\\ \text{-STRESS}\end{bmatrix} \begin{bmatrix}\text{+CONS}\\ \text{-VOICE}\end{bmatrix}_2^2 \begin{bmatrix}\text{+VOC}\\ \text{+VOICE}\end{bmatrix} \\ \begin{bmatrix}\text{--}\\ \text{+CONS}\end{bmatrix} \begin{bmatrix}\text{+VOC}\\ \text{-VOICE}\end{bmatrix} \end{matrix}\right\}$$

4. $$X \longrightarrow \left\{ \begin{array}{lll} [-\text{VOC}] & / \ [+\text{VOC}] \ \text{--} & \# \\ \varnothing & / \ \begin{bmatrix} +\text{VOC} \\ -\text{VOICE} \end{bmatrix} \ \text{--} & \begin{bmatrix} +\text{CONS} \\ -\text{VOICE} \end{bmatrix} \end{array} \right\}$$

5. $$\begin{bmatrix} X \\ \alpha\,\text{STRESS} \end{bmatrix} \begin{bmatrix} X \\ -\alpha\,\text{STRESS} \end{bmatrix} \longrightarrow \begin{bmatrix} X \\ +\text{LONG} \\ +\text{STRESS} \end{bmatrix}$$

The statement of these rules in our notation is the following:

```
RØ   = (ALL R1 R2 R3 R4 R5)

R1   = (+ CNS - CONT - VOICE)
       (+ CONT + VOICE)
       / (+ VOC) -- (+ VOC)

R2   = (ANY R2A R2B)

R2A = (+ VOC)
      (+ STRESS)
      / # (+ CNS) -- (# 1 2 (+ CNS)) (+ VOC) #

R2B = (+ VOC)
      (+ STRESS)
      / # (+ CNS) (+ VOC) (OPT (# 1 2 (+ CNS))) --
```

```
R3   = (ALL R3A R3B R3C)


R3A  = (+ SEG)
       (- VOICE)
       / -- #

R3B  = (+ VOC - STRESS)
       (- VOICE)
       / -- (# 2 2 (+ CNS - VOICE)) (+ VOC + VOICE)

R3C  = (+ CNS)
       (- VOICE)
       / -- (+ VOC - VOICE)

R4   = (ALL R4A R4B)


R4A  = (+ SEG)
       (- VOC - VOICE)
       / (+ VOC) -- #

R4B  = (+ SEG)
       0
       / (+ VOC - VOICE) -- (+ CNS - VOICE)

R5   = (* (! X A STRESS) (= X (- A) STRESS))
       (* 0 (= X + LONG + STRESS))
```

The command

```
WTEST RØ T1
```

will cause the following:

```
TESTING
RØ
((N) # P A P A #)

((N) # P A P A #)
RØ
R1
((N) # P A B A #)
R2
R2A
((N) # P A' B A #)
R3
R3A
((N) # P A' B A- #)
R3B
R3C
((N) # P A' B- A- #)
R4
R4A
R4B
R5
((N) # P A' B- A- #)
```

The initial two lines contain the original test items. Each rule name is printed out before it is applied, and if it is applicable the changed tree is printed. In this case one can easily see that R1, R2A, R3A, and R3C caused all the changes. The final result is also printed at the bottom. The diacritic " ' " indicates stress; "-" indicates devoicing.

A set of rules can be applied to a tree containing more than one syntactic unit. If so, all substructures are transformed first, and the results minus the syntactic categories of the substructure concatenated before transformation on the next level.

Editing and Output Capabilities

The phonological rule testing system has certain editing capabilities built into it as well as having access to the BBN LISP Editor (Bobrow, et al. 1967) both of which can be used to modify the list structures representing rules, trees, and phonemes. Insertions, deletions and replacements and other more sophisticated changes are made easily after a very short learning period. To delete any defined item the user needs type only

```
DEL <Name><Type>
```

where <Name> is the name of a rule, phoneme, or tree and <Type> is either RULE, PHONEME or TREE.

Definitions are printed out by using one of two print commands. The instruction

```
P R1
```

will cause the system to respond with (using the above definition)

```
R1   = (+ CNS - CONT - VOICE)
       (+ CONT + VOICE)
       / (+ VOC) -- (+ VOC)
```

The instruction

```
PR <Type>
```

will cause the entire inventory of the <Type> specified to be printed. Using the definition above, PR RULES would result in the output shown on page 22.

The instruction EDIT calls the BBN LISP editor without leaving the phonological rule tester system. The command

```
EDIT R1 RULE
```

allows one to edit the rule R1.

The use of this editor has been described elsewhere and will not be presented here. (Cf. Bobrow, et.al. 1967)

From the phonological system, a user can write a file containing the data he has generated. By typing

```
SAVE <NAME>
```

a file of that <NAME> will be created and saved. It later can be loaded to initialize the system.

Conclusion

In designing this phonological rule tester, we have endeavored to include capabilities which facilitate stating all the phonological rules to far described in the literature, and some only suggested in private discussions. Accordingly, many rules may be stated in a variety of ways within the limitations imposed by the system, and it is the linguist himself who is forced to set whatever restrictions he feels necessary. Finally, we remark that we have defined, with difficulty, all of the phonological rules found in Chapter 5, Summary of Rules, in The Sound Patterns of English (Chomsky and Halle, 1968) and are currently in the process of verifying the claims made in the text.

BIBLIOGRAPHY

Blair, F. "Programming of the grammar tester", in Specification and Utilization of a transformational grammar. Scientific Report No. 1, IBM Corporation, Yorktown Heights, N. Y. 1966.

Bobrow, D. G., Darley, D. L., Deutsch, L. P., Murphy, D. L. and Teitelman, W. "The BBN 940 LISP System", BBN Scientific Report No. 9, July 1967.

Chomsky, N. Syntactic Structures. 's-Gravenhage: Mouton, The Hague, The Netherlands, 1957.

Chomsky, N. and Halle, M. The Sound Patterns of English. Harper and Row, 1968.

Friedman, J. "A Computer System for Transformational Grammar", Computer Science Report, Stanford University, 1968.

Gross, L. N. On-line programming system user's manual. MTP-59, The MITRE Corporation, Bedford, Massachusetts, March 1967.

Keyser, S. J. and Halle, M. "What we do when we speak", in Recognizing Patterns: Studies in living and automated systems. P. Kolers and M. Eden, Eds., MIT Press, 1968.

Londe, D. and Schoene, W. "TGT: Transformational Grammar Tester", TM-3759/000/00, System Development Corporation, 1967.

Rogers, J. "Some Implication of Two Solutions to a Phonolcgical Problem", IJAL, Vol. 33, No. 3, July 1967, pp 198-205.

Teitelman, W. "Design and Implementation of FLIP, A LISP Format Directed List Processor", BBN Scientific Report No. 10, July 1967.

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Bolt Beranek and Newman Inc 50 Moulton Street Cambridge, Massachusetts 02138	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP

3. REPORT TITLE

A PHONOLOGICAL RULE TESTER

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Scientific. Interim

5. AUTHOR(S) (First name, middle initial, last name)
Daniel G. Bobrow
J. Bruce Fraser

6. REPORT DATE 31 January 1968	7a. TOTAL NO. OF PAGES 31	7b. NO. OF REFS 5

8a. CONTRACT OR GRANT NO. F19628-68-C-0125 ARPA Order No. 627	9a. ORIGINATOR'S REPORT NUMBER(S) BBN Report No. 1589 Scientific Report No. 5
b. PROJECT NO. Task, Work Unit Nos. 8668-00-01	
c. DoD Element 6154501R	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) AFCRL-68-0085
d DoD Subelement N/A	

10. DISTRIBUTION STATEMENT
Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public.

11. SUPPLEMENTARY NOTES This research was sponsored by the Advanced Projects Agency	12. SPONSORING MILITARY ACTIVITY Air Force Cambridge Research Laboratories (CRB) L. G. Hanscom Field Bedford, Massachusetts 01730

13. ABSTRACT

In this paper, we report on the design and implementation of a system to alleviate the problem of rule evaluation for the linguist in the area of phonology. The system permits the user to define, on-line, sets of rules statable within the framework presented in The Sound Patterns of English (N. Chomsky and M. Halle, 1968), to define phonemes as bundles of specified distinctive features, to define data as strings of phonemes with associated grammatical structure, to test the effect of applying rules to the data, and to store both the definitions and results. The system was written in BBN LISP (Bobrow *et al*. 1967) on the Scientific Data System 940 computer. The rule application facility described in detail later was implemented by translating the linguistic rules to rules in FLIP (Teitelman, 1967), a format directed list processor embedded in LISP. This made the system construction easy while providing very sophisticated capabilities for the linguist. The system is designed to be used on-line in interactive fashion, with control returned to the user after each command is executed.

DD FORM 1 NOV 65 1473 (PAGE 1)
S/N 0101-807-6811

Unclassified
Security Classification

A-31409

Unclassified
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
phonology						
rule tester						
linguistics						
transformational grammar						
LISP						
Format directed list processing						
on-line systems						

DD FORM 1 NOV 65 1473 (BACK)
S/N 0101-807-6821

Unclassified
Security Classification

A-31409