XII. LINGUISTICS*

Academic and Research Staff

Prof. R. Jakobson
Prof. J. A. Fodor
Prof. M. Halle
Prof. J. J. Katz
Prof. E. S. Klima
Prof. G. H. Matthews
Prof. C. Watkins
Prof. V. J. Zeps
Dr. S. Bromberger
Dr. J. V. Canfield
Dr. M. F. Garrett
Dr. J. S. Gruber
Dr. S.-Y. Kuroda
Dr. W. K. Percival
Dr. A. Schwartz
Dr. D. E. Walker
J. J. Viertel

Graduate Students

E. W. Browne
R. J. Carter
R. P. G. DeRijk
J. L. Fidelholtz
M. L. Geis
J. W. Harris
P. S. Peters, Jr.
C. B. Qualls
J. R. Ross

A. SURVEY OF LATVIAN MORPHOPHONEMICS

Although the number of alternations encountered in Latvian phonology is rather large, they are determined by a relatively small number of rules.

In this report we shall briefly survey, in the probable order of their application, seven such rules or rule complexes. We shall refer to these as (1) the k/c rule (velars [k, g] are replaced by strident dentals [c, dz] in the environment before front vowels [i, æ]), (2) the i/j rule (diffuse vowels [i, u] are replaced by their corresponding glides [j, w] prevocalically), (3) the n/i rule (a tautosyllabic preconsonantal n is replaced by a diffuse vowel identical in gravity with the preceding vowel), (4) the æ/e rule (æ is narrowed to e before i or j, with or without intervening consonants or glides), (5) metathesis (diphtongs with a compact first vowel [a, æ(e)], metathesize), (6) the V/ϕ rule (a vowel is replaced by zero if the next morpheme starts with a vowel or s, and at the end of a word), and (7) the syncope rule (sequences of identical vowels are replaced by long vowels).

The above rules are applicable to strings assembled in the morphological component; the output of these rules is subject to the application of further rules (mutation of dentals, assimilation, etc.). The more important of these will be discussed at the end of this report.

(1) The k/c rule.

Velars (k, g) are replaced by strident dentals (c, dz) in the environment before front vowels (i, æ):

*This work was supported principally by the U.S. Air Force (Electronics Systems Division) under Contract AF 19(628)-2487; and in part by the Joint Services Electronics Programs (U.S. Army, U.S. Navy, and U.S. Air Force) under Contract DA 36-039-AMC-03200(E), the National Science Foundation (Grant GK-835), the National Institutes of Health (Grant 2 PO1 MH-04737-06), and the National Aeronautics and Space Administration (Grant NsG-496).

k, g > c, dz/_____ FV

Typical examples of this alternation are pairs like ruoka 'hand' and its diminutive ruociņa, aûgu 'I grow' and the corresponding causative aûdzinu, druska 'bit' and its diminutive drusciņa, and ganu 'I herd' with the related dzænu 'I drive'.

Since the k/c rule is very early within the morphophonemics of Latvian, its effects are often overlaid with those of other rules, so that apparent exceptions occur in terminal strings.

First of all, the effects of the subsequent V/ϕ rule (rule 6 in this series), which deletes vowels prevocalically and finally, may obscure the conditions that lead to the k/c alternation: aûdz 'you grow' and aûg 'he grows' (from aûg + æ + i and aûg + a, respectively), rædzu 'I see' (from rægæ + au), saucu 'I call(ed)' (from saukæ + au [Pres.] or saukæ + a + au [Past]).

Second, in cases where metathesis (rule 5) takes place in addition to vowel truncation (rule 6), it may appear that the environment calls for a k/c alternation, where non occurs: vilki 'wolves' (from vilk + ai > [met.] vilk + ia > [V/ϕ] vilk + i), saki 'you say' (from sak + a + æi > [met.] sak + a + ia > [V/ϕ] sak + i).

Third, the k/c alternation, seemingly implicit in morphological assemblies such as skæl + ti 'to split', may be superseded by the operation of another rule, e.g., the k/ķ alternation in the environment s _____ FV (precise statement elsewhere): šķelt.

Fourth, the k/c alternation may be ruled out from some (foreign) lexical items: kinō 'movies'.

As far as we have been able to ascertain, the k/c rule is not ordered with respect to rules (2), (3), and (4); it must, however, precede rules (5) and (6), as shown by examples aûdz and vilki, above.

(2) The i/j rule.

All native instances of j and v point to their being prevocalic realizations of the vowels i and u, respectively, e.g., šuvu 'I sewed' and šūt 'to sew', lija 'it rained' and līt 'to rain', skreju 'I run' and skriet 'to run'.

Accordingly, no instances of v and j appear in the lexical core. All instances of v and j, at least in the basic morphemes, are due to the glide rule, which states, informally, that

i, u > j, w/_____ V

or i and u are replaced by their respective glides before vowels (and a late rule then carries w to v).

While the rule must be stated in the entirely general form above, there are a number of environments from which it has been specifically exempted. In particular, the

rule does not apply to any segment that precedes an identical vowel (i.e., the rule does not apply to the first i in ii or to the first u in uu, and Latvian does not have the sequences ji and vu, except under very special conditions): iilæn + a + s > īlæns 'awl' (not *jilæns) and uudæn + i + s > ūdens 'water' (not *vudens). The rule does not apply to an i before a front vowel in the next morpheme: saaksi + æi > [met.] saaksi + iæ > [V/ϕ] saaks + i > [sync.] sāksi 'you will begin' (and not *sāksi from *sāksj + æi). Nor does the rule apply to an u that follows a morpheme-initial consonant cluster ending in other than a dental obstruent: tvans 'carbon monoxide' but puika 'boy', muita 'customs', kuilis 'boar'.

Further exceptions are either lexical (tuaregs 'Tuareg', diēta 'diet'), or appear in the lexicon with a v and j already specified as non-vocalic: Vladivostoka, jipts colloq. 'poison'.

The i/j rule is not ordered with respect to rules (1), (3), and (4). It must, however, precede metathesis (5) – cf. java 'fluid mixture' and ieva 'bird-cherry' from iau + aa + i and æiu + aa + i, respectively. The i/j rule must likewise precede the vowel truncation rule (6) and the syncope rule (7) – (in order to yield lija 'it rained', the underlying lii + aa may not be replaced by *li + aa or *lī + aa).

(3) The n/i rule.

The alternations covered by this rule involve the conversion of a tautosyllabic preconsonantal or final n into a high vowel identical in gravity with the preceding vowel. Thus, unC is converted into uuC, ænC into æiC, and so on. The form of the rule is, informally:

$$n > \begin{Bmatrix} i/FV \\ u/BV \end{Bmatrix} \_\_\_\_ \begin{Bmatrix} C \\ \# \end{Bmatrix}$$

None of the n/vowel alternations result in sequences that are retained in terminal strings. Some are further subject to the syncope rule: gin + ti > (k/c) dzin + ti > (n/i) dzii + ti > (V/ϕ) dzii + t > (sync.) dzīt 'to drive', iunt + a > (i/j) junt + a > (n/i) juut + a > (V/ϕ) juut > (sync.) jūt 'he feels', krint + a > (n/i) kriit + a > (V/ϕ) kriit > (sync.) krīt 'he falls', lind + aa > (n/i) liid + aa > (V/ϕ) liid + a > (sync.) līda 'he crawled'.

Other intermediate strings (and these constitute the majority) are transposed by the metathesis rule: zanb + a + s > (n/i) zaub + a + s > (met) zuab + a + s > (V/ϕ) zuabs > (other rules) zuops 'tooth', lænk + ti > (n/i) læik + ti > (æ/e) leik + ti > (met.) liek + ti > (V/ϕ) liekt 'to bend', prant + a > (n/i) praut + a > (met.) pruat + a > (V/ϕ) pruat > (assim.) pruot 'he knows how', brænd + n + a > (n/i) bræid + n + a > (æ/e) breid + n + a > (met.) bried + n + a > (V/ϕ and other) brien 'he wades'.

The n/i rule (3) has to precede the æ/e rule (4), if the underlying form for Asf zæm + æn 'earth' is correct. The form zæm + æn should first go to zæm + æi, then narrow to

zem + ei, metathesize to zem + ie and truncate to the correct zemi. Since the narrowing is caused by i, it follows that the n/i rule must precede the æ/e rule. [There are other possible explanations for the above; these, however, seem less attractive.] The n/i rule (3) certainly has to precede metathesis (5), as already shown.

Exceptions to the n/i rule are numerous, and all entirely lexical (i.e., there is no subclass of morphemes, foreign words aside, that consistently retain n): rinda 'row', dzintars 'amber', censties 'to strive', tænkas 'gossip', anglis 'Englishman', and many more.

(4) The æ/e rule.

The narrowing of æ to e takes place before i or j, with or without intervening consonants or glides. Typical are examples like mîeta 'maid', leja 'valley', slēpju 'I hide', celu 'I lift' (from kæ̀li + au). The narrowing, furthermore, is "contagious" insofar as any number of vowels will be narrowed, so long as an a or u doesn't intervene: æcâ̂tu 'I would harrow' and ecêsi 'you will harrow'.

Apparent discrepancies are of two kinds. In one case, it appears as if the conditions have been met, but no narrowing has taken place. The original conditions have been most typically obscured by metathesis and truncation, e.g., dâ̂li (from dæâ̂el + ai) 'sons'. There are, however, other exceptions, e.g., verbs in -in-: vēdinât 'to air'.

Conversely, it may appear that the conditions have not been met, yet the narrowing has occurred. Here we typically have to do with medial or terminal vowel truncation, e.g., mest 'to throw' (from mæt + ti). Except under special conditions (æixe, xærmanis 'Eiche, Hermanis'), e in loanwords is narrow: 'leta 'counter', texnika 'technology', vetō 'veto'.

As already shown, the æ/e rule must precede metathesis, and follow the n/i rule.

(5) Metathesis.

In Latvian phonology, metathesis plays a central role. It is unconditioned, i.e., the rule applies to all appropriate sequences regardless of environment, unless specifically blocked. Subject to metathesis are ai, au, au from an, æi, æi from æn, and æu, whereby æi from æn and au from an metathesize invariably.

Exempt from metathesis are: ai in root syllables: maize 'bread', laiva 'boat'; au in root syllables unless before a root-final vowel: saule 'sun' but guovs 'cow' (from gauu + i + s); archaic imperatives eima 'let's go' and eita 'go!'; a small number of highly frequent stems with æi (and their derivatives: beigt 'to end,' beigas 'end', beigts 'dead', teikt 'to say', teika 'legend', teicams 'praiseworthy', etc.; and late loans: streiks 'strike'.

Examples of metathesis include skræi + n + a > skrien 'he runs' vs. skrēja 'he ran', lænk + a > liek 'he puts', dæu + d + a > duod 'he gives' vs. deva 'he gave', vilk + ai >

(met.) vilk + ia > > (V/ϕ) vilki 'wolves', skaties 'you look' (from skat + a + æisi), lank + a + s > (met.) lauk + a + s > (V/ϕ and other) luoks 'bow', sit + a + au > (met.) sit + a + ua > (V/ϕ) situ 'I hit (past)'.

The place of metathesis in the sequence of rules is perhaps best motivated of all. Rules (1) through (4) must precede it (see above), and the V/ϕ rule must follow it (see below).

(6) The V/ϕ rule.

Terminal strings in Latvian may not have numerous successive vowels; to the degree that they have not been converted into glides, excess vowels are truncated. Vowels must be truncated if the next morpheme (from a phonological point of view – i.e., a stretch of segments flanked by pluses) starts with a vowel or s and at the end of the word:

$$V \rightarrow \phi \,/\, \_\_\_\_ \begin{cases} + \begin{cases} V \\ s \end{cases} \\ \neq \end{cases}$$

Accordingly, the reflex of an assembly of the type rægǣ + au 'I see' is rædzǣ + au (k/c rule) → rædzǣ + ua (metathesis) → rædzu (truncation); sāki 'you began', is derived from saak + a + æi (metathesis) → saak + a + iæ (truncation) → saak + i (syncope) → sāki. Truncation before + s is illustrated by vilks 'wolf' from vilk + a + s.

Truncation must follow metathesis, as the examples rædzu and sāki illustrate. It must precede syncope (see below).

(7) Syncope.

The syncope rule contracts sequences of identical vowels into single long vowels; lii + ti → līt 'to rain'. Its place after the truncation rule is shown by the following example: skat + aa → skat + a 'he views' (truncation), i.e., the syncope rule changing aa to ā (as in skatām 'we view') may not apply prior to the truncation rule.

The rules just cited imply obviously appropriate underlying representations so that the attested phonetic reflexes may be derived. The underlying strings in the major flexional categories of Latvian are of special interest and will be reviewed below. Rules other than the seven just discussed apply as well; they do not, however, affect our conception of the underlying forms.

We illustrate the declension with the assemblies and phonetic realizations of vilk m. a-stem 'wolf', gulb m. i-stem 'swan', lap f. a-stem 'leaf' and zæm f. æ-stem 'earth'.

Nsm	vilk + a + s	vilks	gulb + is	gulbis
Gsm	vilk + a + aa	vilka	gulb + iaa	gulbja
Dsm	vilk + ami	vilkam	gulb + imi	gulbim
Asm	vilk + an	vilku	gulb + in	gulbi
Lsm	vilk + aaia	vilkā	gulb + iiia	gulbī

Npm	vilk + ai	vilki	gulb + iai	gulbji
Gpm	vilk + au	vilku	gulb + iau	gulbju
Dpm	vilk + aimi	vilkiem	gulb + iaimi	gulbjiem
Apm	vilk + au + s	vilkus	gulb + iau + s	gulbjus
Lpm	vilk + ausi	vilkuos	gulb + iausi	gulbjuos
Nsf	lap + aa + i	lapa	zæm + ææ + i	zeme
Gsf	lap + aa + si	lapas	zæm + ææ + si	zemes
Dsf	lap + aia	lapaj	zæm + æia	zemej
Asf	lap + an	lapu	zæm + æn	zemi
Lsf	lap + aaia	lapā	zæm + ææia	zemē
NApf	lap + aa + si	lapas	zæm + ææ + si	zemes
Gpf	lap + au	lapu	zæm + iau	zemju
Dpf	lap + aami	lapām	zæm + ææmi	zemēm
Lpf	lap + aasi	lapās	zæm + ææsi	zemēs

The indefinite adjective is declined like the corresponding masc. and fem. a-stems: Nsm mazs 'small', Gsm maza, etc., Nsf maza, etc.

The definite adjective and the pronoun differ from the noun in one of two ways – either an element ai is infixed (in the Nsm, D, and L forms of the adjective, and in the L forms of the pronoun), or the morpheme containing the thematic a-vowel is exempt from the truncation rule. Paradigms of mazajs 'the small (one)' and tas 'that' follow.

Nsm	maz + aia + s	mazajs	t + a + s	tas
Gsm	maz + a + aa	mazā	t + a + aa	tā
Dsm	maz + aiami	mazajam	t + ami	tam
Asmf	maz + an	mazuo	t + an	tuo
Lsmf	maz + aiaaia	mazajā	t + aiaaia	tajā
Nsf	maz + aa + i	mazā	t + aa + i	tā
Gsf	maz + aa + si	mazās	t + aa + si	tās
Dsf	maz + aiaia	mazajaj	t + aia	taj
Asf as Asm, Lsf as Lsm				
Npm	maz + ai	mazie	t + ai	tie
Gpm	maz + au	mazuo	t + au	tuo
Dpm	maz + aiaimi	mazajiem	t + aimi	tiem
Apm	maz + au + si	mazuos	t + au + s	tuos
Lpm	maz + auausi	mazajuos	t + aiausi	tajuos

NApf	maz + aa + si	mazās	t + aa + si	tās
Gpf as Gpm				
Dpf	maz + aiaami	mazajām	t + aami	tām
Lpf	maz + aiaasi	mazajās	t + aiaasi	tajās

In the conjugation, first note the assemblies and realizations of the future tense and its reflexive for the verb likt 'to put':

1s	liksi + au	likšu	liksi + ausi	likšuos
2s	liksi + æi	liksi	liksi + æisi	liksies
1p	liksi + mi	liksim	liksi + mi + æisi	liksimies
2p	liksi + ti	liksit	liksi + ti + æisi	liksities
3	liksi	liks	liksi + æisi	liksies

The reflexive morpheme has two allomorphs. The allomorph + æisi occurs after Ci, the allomorph si (without +) occurs elsewhere. The first person sg. morpheme is au, the second person sg. morpheme is generally æi, but in the present tense non-reflexive assemblies of a number of verbs, the second person sg. is introduced with a morpheme boundary as æ + i. The past tense morpheme is realized as aa, the present tense morpheme as aa or a depending on the stem-clan. Prior to the application of any morphophonemic rules, one a is deleted from the singular assembly (i.e., mæt + a + æ + i, underlying met 'you throw', is adjusted to mæt + æ + i, and saak + aa + æi, underlying sāki 'you began', is adjusted to saak + a + æi). This adjustment is crucial if the k/c rule, the i/j rule, the æ/e rule and the V/ϕ rule are to operate properly. A few paradigms follow, already adjusted as to the extra a. The stems are mæt ϕ-pres. stem 'throw', lænk ϕ-pres. stem 'put', sak ī-past stem 'say'

Present

1s	mæt + au	mætu	lænk + au	lieku	sak + a + au	saku
2s	mæt + æ + i	met	lænk + æ + i	liec	sak + a + æi	saki
1p	mæt + a + mi	mætam	lænk + a + mi	liekam	sak + aa + mi	sakām
2p	mæt + a + ti	mætat	lænk + a + ti	liekat	sak + aa + ti	sakāt
3	mæt + a	mæt	lænk + a	liek	sak + aa	saka

In the past tense, the stem allomorphs met, lik, and sakīi are morphologically conditioned:

1s	met + a + au	metu	lik + a + au	liku
2s	met + a + æi	meti	lik + a + æi	liki
1p	met + aa + mi	metām	lik + aa + mi	likām
2p	met + aa + ti	metāt	lik + aa + ti	likāt
3	met + aa	meta	lik + aa	lika

1s	sakīi + a + au	sacīju
2s	sakīi + a + æi	sacīji
1p	sakīi + aa + mi	sacījām
2p	sakīi + aa + ti	sacījāt
3	sakīi + aa	sacīja

In addition to the seven rules discussed at the beginning of this report, a number of other rules apply as well. These, however, are all sufficiently close to the phonetic surface, that they present no theoretical or practical difficulties. The most important among these are the following forms

(a) The t/s alternation.

Dental stops (t, d) are replaced by spirants (s, z) in the environment before other dental stops and before j.

t, d > s, z/_____ t, d, j

e.g., mæt + ti > (æ/e) met + ti > (V/ϕ) met + t (t/s) mest 'to throw'. In most cases, the immediate reflex of t and d (i.e., s, z) is subject to further rules: the mutation of dentals to palatals (b) – Dat. sg. mædi + ami > (i/j) mædj + ami > (æ/e) medj + ami > (V/ϕ) medj + am > (t/s) mezj + am > (palat.) mežj + am > (j/ϕ) mežam 'forest'; assimilation in voicing (d) – væd + ti > (æ/e) ved + ti > (V/ϕ) ved + t > (t/s) vez + t > (assim.) vest 'to lead'; or loss (not discussed in this report) – kliid + tt + a > (V/ϕ) kliid + tt > (sync.) klīd + tt > (t/s, across a plus) klīz + tt > (t/s) klīz + st > (assim.) klīs + st > (loss[or syncope?]) klīst 'he strays'.

It is essential that the t/s rule precede the mutation of dentals – cf. zutis 'eel' and Gs zuša (from zut + iaa > (i/j) zut + jaa > (V/ϕ) zut + ja > (t/s) zus + ja > (palat.) zuš + ja > (j/ϕ) zuša.

(b) The mutation of dentals.

Dental consonants are replaced by corresponding palatals in the environment before j:

c, dz, s, z, n, l, r > č, dž, š, ž, ņ, ļ, ŗ/_____ j, e.g., laak + iaa > (k/c) laac + iaa > (i/j) laac + jaa > (V/ϕ) laac + ja > (sync.) lāc + ja > (palat.) lāč + ja > (j/ϕ) lāča 'bear Gen sg.'

The mutation rule may be considerably more general than suggested above and may include the environment "after j" as well (vējš 'wind' from uæəi + a + s) and the environment before other palatals (šķelt 'to split' from skæl + ti). There are, however, some minor difficulties in the formulation of a more general rule, among them the apparent counterexamples, such as maijs 'May' (not *maijš) and runča Gs 'tomcat' (not *runča).

(c) The j/ϕ alternation.

In a number of environments, a j is lost. These include the environment after long vowels at the end of polysyllabic words, e.g., mazgāi + a > (i/j) mazgāj + a > (V/ϕ) mazgāj > (j/ϕ) mazgā 'he washes'; as well as the environment after a palatal (the latter generated by the previous rule): naz + iai > (i/j) naz + jai > (met.) naz + jia > (V/ϕ) naz + ji > (palat.) naž + ji > (j/ϕ) nažji 'knives'.

It is not clear whether the j/ϕ loss is connected with the loss of other segments, e.g., d before n: brænd + n + a > (n/i) bræid + n + a > (æ/e) breid + n + a > (met.) bried + n + a (V/ϕ) bried + n > (d/ϕ) brien 'he wades'.

(d) Assimilation in voicing.

Obstruent clusters are subject to a regressive assimilation in voicing, e.g., zirg + a + s > (V/ϕ) zirg + s > (assim.) zirks 'horse'. This assimilation must follow j-loss: mædi + a + s > (i/j) mædj + a + s > (æ/e) medj + a + s > (V/ϕ) medj + s > (t/s) mezj + s > (palat.) mežj + š > (j/ϕ) mežš > (voicing) mešš 'forest'.

M. Halle, V. J. Zeps

B. ON THE METRICS OF PRE-ISLAMIC ARABIC POETRY

The purpose of the following note is to give wider currency to certain facts of great linguistic interest which because of the present somewhat artificial organization of scholarly publication are likely to escape the attention of all but a very small number of linguists. The note is based on G. Weil's article ʿArūḍ in the Encyclopedia of Islam, I (Leiden, 1960), pp. 667-677.

The term ʿarūḍ is used by the native Arab grammarians to designate the "science of the rules by means of which one distinguishes correct metres from faulty ones in ancient (pre-Islamic – M. H.) poetry." (667). In line with this conception of the primary objectives of the science of prosody, the ancient grammarians distinguished between uṣūl, which refers to the abstract underlying patterns, and furūʿ, which refers to the set of verse types by means of which the abstract patterns may be actualized. If this distinction were to be applied in the study of English prosody, basic verse patterns such as "iambic pentameter" or "trochaic dimeter" would belong to uṣūl, whereas the rules for actualizing the iambic pentameter – e.g., the list of the "allowable deviations from the basic iambic pentameter" – would belong to the furūʿ.

In the present note we restrict our attention to the rules that establish the 16 uṣūl patterns recognized by the native prosodists.

Rule 1. The verse line is composed of two identical hemistichs (miṣrāʿ pl. maṣāriʿ).[1]

Rule 2. A hemistich contains either three or four pegs (watid pl. awtad).

Rule 3. In a given hemistich each peg is preceded by the same number of cord units (sabab pl. asbab). This number may vary between one and three.

Rule 4. In hemistichs with four pegs no more than two cord units are admitted before a peg.

Rule 5. In hemistichs with three pegs, no less than two cord units must precede each peg.

If we represent a peg by P and a cord unit by C, we obtain from rules 2-5 the following four abstract patterns, which we designate here by the numerals with which they are labelled in the Arab treatises:

(1)

CPCPCPCP	V
CCPCCPCCPCCP	I
CCPCCPCCP	III
CCCPCCCPCCCP	II

Rule 6. A peg is composed of a weak position and strong position.

In a regular line of verse the weak position is occupied by a short unstressed syllable and the strong position is occupied by a long stressed syllable. When the weak position precedes the strong position we have an iambic peg (watid maǰmūᶜ); when the strong position precedes the weak position, we have a trochaic peg (watid mafrūk). The occurrence of trochaic pegs is severely restricted (see rule 7 below).

A cord unit is normally occupied by a single syllable whose quantity and stress are apparently free.

Rule 7. In three-peg hemistichs with two intervening cord units (pattern III in (1)) the last peg may be either iambic or trochaic; otherwise trochaic pegs are not admitted, and only iambic pegs are found.

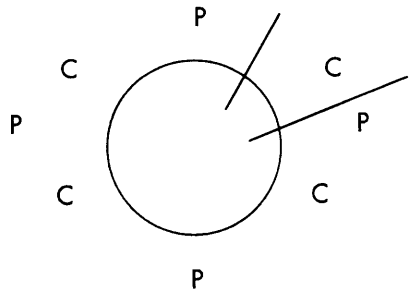
Thus, if we assume that P in (1) stands for an iambic peg, and let Q stand for a trochaic peg, we may think of rule 7 as adding a fifth pattern to the four cited m (1):

(2) CCPCCPCCQ (IV)

Rule 8. Subject to the restrictions below (see rules 9-11) the set of admissible hemistichs is given by the strings formed by cyclical permutation from the five patterns in (1) and (2).

Rule 8 expresses the main constitutive principle of Arabic. The founder of Arabic prosody, the eighth century scholar, Al-Xalīl, expressed this fact by representing the five patterns in (1) and (2) as circles. Thus, for instance, pattern V of (1) above was represented by Al-Xalīl in a form that is essentially equivalent to (3)

(3)



where the two lines intersecting the circle indicate the two possible terminal points of the hemistich. In view of this the five basic patterns of (1) and (2) are referred to traditionally as circles. This term will also be utilized below.

Assuming that the metrical entities are to be read in clockwise order, (3) can be taken as representing the two hemistichs which Al-Xalīl designated as

(4)

mutadārik	CPCPCPCP
mutaḳārib	PCPCPCPC

It is obvious that from the pattern of circle V the strings in (4) are the only ones that can be generated by cyclical permutation. From the pattern of circle III in (1) three distinct strings can be generated (as before the names on the left are those used by Al-Xalīl):

(5)

rajaz	CCPCCPCCP
hazaj	PCCPCCPCC
ramal	CPCCPCCPC

Similarly, from the pattern of circle I in (1) only three distinct strings can be generated:

(6)

CCPCCPCCPCCP
PCCPCCPCCPCC
CPCCPCCPCCPC

These, however, are subject to a special adjustment rule:

Rule 9. In hemistichs of the first circle delete the cord unit following even numbered pegs, if the line begins with the sequence CP; otherwise, delete the cord unit following odd numbered pegs.

Rule 9 yields then the three attested strings of circle (I):

(7)

basiṭ	CCPCPCCPCP
ṭawīl	PCPCCPCPCC
madīd	CPCCPCPCCP

Rule 10. A sequence of three cord units functions in the permutation as a single unit.

Rule 10 accounts for the fact that in circle II of (1) only two strings are generated:

(8)

kāmil	CCCPCCCPCCP
wāfir	PCCCPCCCPCCC

Rule 11a. A trochaic peg cannot begin a hemistich.

Rule 11b. A hemistich may not end in a trochaic peg followed by one or more cord units.

These two rules affect only strings in circle IV. Rule 11a excludes the string

QCCPCCPCC

Rule 11b excludes the strings

PCCPCCQCC
CPCCPCCQC

The remaining six strings that may be generated by cyclical permutation from circle IV are all admissible:

(9)

sarīᶜ	CCPCCPCCQ
mu̯jtaΘΘ	CQCCPCCPC
muḳtaḍab	CCQCCPCCP
muḍāriᶜ	PCCQCCPCC
xafīf	CPCCQCCPC
munsariḥ	CCPCCQCCP

It must be noted that rules 9, 10, 11 have a rather unmotivated appearance in the form in which they are given above. This suggests that something essential has been missed here.

The preceding discussion differs from the traditional treatment in that it dispenses with the entities foot (ju̯zᶜ) and mora, which play a prominent role in the traditional discussions. We have found no use for these entities in our description, and at the present state of our understanding we are inclined to believe that these entities appear in the traditional discussions because of certain features of the symbolic apparatus that is utilized in the traditional discussion, rather than because of properties of the subject matter.

M. Halle

Footnote

1. This term is reminiscent of the Germanic "stave".

C. ON THE INTERSECTION OF REGULAR LANGUAGES AND LANGUAGES GENERATED BY TRANSFORMATIONAL GRAMMARS

1. Introduction

The study[1] of transformational grammars presented in this report is based on the definition of these objects which is forthcoming by Stanley Peters and Robert Ritchie. This definition is in private circulation among workers in the field, and will not be stated here, as it would be twenty pages in length. We shall, however, briefly recapitulate their definition for the present purposes.

A transformational grammar, TG, consists of two parts. The first is a phase-structure grammar which generates bracketed strings of terminal symbols. The second is a sequence of transformations. A transformation is an ordered pair, the first member of which is a structural condition, and the second member of which is a sequence of operations. A structural condition, SC, is a Boolian combination of three predicates on a given factorization of a bracketed string: (i) one sort of predicate says that a certain factor or concatenation of factors is enclosed in a bracket of a certain type; (ii) the second sort says that the interior of a certain factor in the factorization is identical with the interior of another factor; and (iii) the third sort says that the debracketization of a certain factor is identical to a given terminal string. The operations performed by a transformation are of three elementary kinds: (i) the operation of deleting a certain factor; (ii) the operation of replacing a certain factor by a sequence of other factors; and (iii) the operation of adjoining a sequence of factors to a certain factor. The sequence of transformations in the transformational component of the grammar apply by convention to the innermost S-bracket of a string 'working out' to the outermost.

The principle of recoverability of deletions is a constraint on the operations that transformations in the transformational component of the grammar may perform. The principle requires, in essence, that if the i^{th} factor is deleted or is substituted for by the j^{th}-k^{th} factors, then, either the i^{th} factor must be identical to some other factor which is not itself deleted or substituted for by the transformation, or else it must be identical to one of a fixed finite set of terminal strings. It was thought (see e.g., Katz and Postal, An Integrated Theory of Linguistic Descriptions) that placing this constraint on CF-based TG's would make any language generated by such a device fully recursive. Peters and Ritchie have shown that context-sensitive based TG's which meet the condition on recoverability of deletions generate a class of languages which (modulo Church's thesis) is coextensive with the class of all recursively enumerable languages.

We shall be concerned with the implications consequent upon a positive solution to the intersection problem (IP) for transformational grammars (TG). The IP is as follows: Given a context-free based TG, G_1, which meets the condition on recoverability of deletions, and a regular language, H, does there exist a TG, G_2, which is CF-based and

meets the condition on recoverability, such that $L(G_2) = L(G_1) \cap H$?

Note first that a negative solution to this problem implies that the intersection of a CF-based recoverable TG with a CF, context-sensitive or TG language is not generated by another CF-based recoverable TG. As we shall see, however, the interest of IP rests only peripherally on the implications of its negative solution. In essence, a positive solution to IP means that there are several linguistically important properties of CF-based recoverable TG's that hold.

We shall consider TG's that meet the condition on recoverability of deletions and have a CF base only.

2. The Filter Problem

Chomsky[2] states two functions performed by transformations in the construction of grammars for natural languages: the first is "to convert an abstract deep structure that expresses the content of a sentence into a fairly concrete surface structure that indicates its form."[3]; and the second is to act as filters on strings generated by the context-free base allowing only a subset of those strings to qualify as deep structures for sentences grammatical in the language.[4] The formal device that he suggests for accomplishing this is that if some obligatory transformation which should have applied did not because its structural condition was not fulfilled, then a certain symbol called 'sentence boundary' is left in the interior of the string. By convention, then, we accept no strings in which sentence boundaries lie in their interiors.

It is evident that we can state the result of this convention formally by intersecting the transformational language under consideration with a regular language, H, which consists of all those strings over the terminal vocabulary which do not have sentence boundaries occurring in their interiors.

Let us formalize the discussion above.

Definition 1: A filter is a transformation that must apply in each transformational cycle in the derivation of a string in order for that string to be considered an element of the language under consideration. (Notice that this is not a precise statement of what we need linguistically, for, in fact, transformations act as filters in natural languages usually when a certain part of their structural condition is fulfilled, but another part, e.g., an identity condition, is not. It is immediately evident, however, that we can accomplish this result by breaking the transformation in question down into two transformations which perform the same operation and such that one of them is a filter by the above definition.)

Definition 2: The filter problem (FP) is as follows: Is the class of languages generated by TG's with filters larger than the class of TG languages?

LEMMA 1: A positive solution to IP implies a negative solution to FP. (IP $\Longrightarrow$ -FP)

Proof: Let G_1 be a TG, a subsequence of the transformations of which, $T_{i_1} \ldots T_{i_k}$, are filters. Form a TG, G_2, without filters as follows: the base components are the same

except that new symbols $\beta_{i_1} \ldots \beta_{i_k}$ are generated on the right-hand of each S-phrase (i.e., if $S \rightarrow \omega_j$ is a rule of P_1, then $S \rightarrow \omega_j\beta_{i_1} \ldots \beta_{i_k}$ is a rule of P_2). The transformations are the same, except that for each filter T_{i_j} we add onto its SC the condition that it senses the symbol β_{i_j}, and to its operation an erasure of β_{i_j}. Let H be the regular language over the enlarged terminal vocabulary consisting of those strings in which none of the symbols $\beta_{i_1} \ldots \beta_{i_j}$ ever occurs. It is evident that $L(G_1) = L(G_2) \cap H$.

Notice that by the lemma, then, a positive solution to IP is the 'happy' solution linguistically, in the sense that the linguist is not allowing an increase in the class of structures which he is considering as potential grammars by the use of filters. On the other hand, even a constructive solution to IP, i.e., one such that we give a procedure for constructing G_2, given G_1, and H, would not mean that we could eliminate filters from use in the present framework of a transformational study of language, for the strong generative capacity would evidently differ between G_1 and G_2. And it is linguistically important that the terminal strings have the correct surface structure, e.g., for input to the stress assignment rules of the phonological component.

3. The Optional-Obligatory Problem (OOP)

In the definition of TG by Peters and Ritchie, a transformation must apply if its SC is satisfied by some analysis of the string. We have shown above that we can restrict our language to those strings in which a certain transformation, in fact applied in each cycle, modulo the IP. A natural question then concerns the class of languages generated by TG's in which one or more transformation applies optionally. We formalize this notion and the OOP as follows.

Definition 3: A transformation is optional when its application implies its SC has been met. (It is obligatory when it applies iff its SC has been met.)

Definition 4: The optional-obligatory problem is as follows: Does the class of languages generated by TG's in which one or more transformation is optional properly include the class of languages generated by TG's, all of whose transformations are obligatory?

LEMMA 2: A positive solution to IP implies a negative solution to OOP. (IP $\Longrightarrow$ -OOP.)

Proof: We shall consider a TG, G_1, in which only one transformation is optional, and it will be evident that the method of proof can be generalized to the case in which an arbitrary subsequence of transformations of G_1, including all of them, is optional.

Let $G_1 = (P, (T_1 \ldots T_k))$, where T_i is optional. Let $G_2 = (P', (T_1 \ldots T_i' \ldots T_k))$, where P' is like P, except that for every rule in P of the form $S \rightarrow \omega_j$ introduce in addition a rule in P' $S \rightarrow \omega_j\beta_i$; and let T_i' be like T_i, except that in the SC of T_i' we add the condition that the symbol β_i appear on the right and be erased when T_i' applies. Let H be all strings over the enlarged terminal vocabulary in which β_i never appears. We

claim that $L(G_1) = L(G_2) \cap H$. This can be seen by checking cases.

Case I: T_i applied in a certain cycle in G_1. In this case, the symbol β_i was generated by P' and T'_i applied in G_2.

Case II: The SC of T_i was satisfied in a certain cycle, but T_i by option failed to apply. Corresponding to this in G_2, all the SC of T'_i was satisfied, except that the symbol β_i failed to appear as it did not happen to be generated in that cycle. Thus, T'_i failed to apply as its SC was not completely fulfilled.

Case III: The SC of T_i was not fulfilled, and therefore it did not apply in G_1. Then, in G_2 either β_i was generated by P' or it was not. In the latter case, nothing is altered. In the former case, the symbol β_i will not be erased by T'_i and will therefore appear as a terminal symbol; however, this string will not appear in the intersection of $L(G_2)$ and H, since it does not lie in H.

In fact, it would appear that we can arrive at a negative solution to OOP directly without the necessity of a positive solution to IP. In the proof above, add a transformation T_{k+1} which applies in the environment $X\beta_i Y$, and which deletes β_i. Then β_i will never be left unerased in any string generated by G_2, and it is evident that $L(G_1) = L(G_2)$. Thus we can always transfer optionality in the transformations into optionality in the base.

4. The Outermost S-Bracket Problem

According to the definition of Peters and Ritchie, it is impossible for the transformations to 'know' in which S-cycle they are operating in a string generated from the base; in particular, they cannot 'know' when they are in the final S-cycle. From a linguistic point of view this is a deficiency, for there is good evidence that in addition to the cyclical transformations, i.e., the transformations that apply on each S-level from the innermost one out, there must also be a set of postcyclic transforms that apply only at the outermost S-bracket. Furthermore, these postcyclic transformations are interspersed in order among the cyclic transformations in application at the outermost S-bracket.

Definition 5: A postcyclic transformational grammar is a transformational grammar in which a subsequence, $T_{i_1} \ldots T_{i_k}$, of the transformations applies only at the outermost S-bracket if they apply at all to a string.

Definition 6: The outermost S-bracket problem is as follows: Is the class of postcyclic transformational languages properly larger than the class of transformational languages?

LEMMA 3: Let G_1 be a postcyclic TG, then a positive solution to IP implies that there exists a TG, G_2, such that $y \in L(G_1)$ iff $yt \in L(G_2)$, where t is a single symbol not in V_T.

Proof: Let $T_{i_1} \ldots T_{i_k}$ be the postcyclic transformations of G_1. Form G_2 by putting an

additional rule $S \rightarrow \omega_i t$ in P_2 whenever the rule $S \rightarrow \omega_i$ was in P_1. Add the postcyclic transformations to the transformational component of G_2, preserving the relative order of all transformations as they applied on the last cycle in G_1, except that the structural condition of the postcyclic transformations are altered so that they will apply only in the environment Xt in addition to the rest of their SC's. Now, let H be the regular language consisting of all strings over the enlarged terminal vocabulary in which the symbol 't' occurs at the end and only at the end of each string.

The claim is that $y \in L(G_1)$ iff $yt \in L(G_2) \cap H$. For let $y \in L(G_1)$, where y was derived transformationally from the string y' which was generated by the base P_1. We then generate (y')t in the base P_2, and exactly the same transformations which applied to y' will apply to (y')t, including the postcyclic transformations that will apply in the outermost S-bracket of (y')t because the presence of 't' triggers their application in G_2 as we have defined it above. Also, $yt \in H$ by the definition of H. Now let $x \in L(G_2) \cap H$. Since $x \in H$, we know x = yt, where y contains no occurrences of the symbol 't'. Let yt be transformationally derived from the base structure (y')t. We know that $y' \in L(P_1)$ by the construction of P_2. And by the definition of the transformational component of G_2, we know that the same sequence of transformations will apply to (y')t in G_2 as will apply to y' in G_1.

This lemma can be strengthened by using the same methods of proof. That is, all that we have required of the postcyclic transformations in that they apply if they can, but only to the outermost S-bracket. Suppose we require, however, that in addition all the postcyclic transformations must apply in the last cycle (i.e., a string will be in the language iff all of the postcyclic transformations did in fact apply in the last cycle). We might call these postcyclic filters, i.e., transformations which say that the final form of the string must meet such-and-such structural conditions and must have had such-and-such operations applied to it in the last cycle.

The question, as usual, is whether postcyclic filter grammars are essentially stronger than grammars defined by Peters and Ritchie. And the answer, as usual, is that a positive solution to IP implies a negative solution to this problem. The proof utilizes the same methods as the proofs of Lemmas 1 and 3, and may be left as an exercise.

It may be of interest to note that by using intersection with a regular language we could demand that certain transformations apply, e.g., to all S-brackets nested, say, 17 levels below the outermost S-brackets; or that we can demand that transformation T_i apply exactly 5 cycles later than transformation T_j; etc. We shall not prove these assertions, although the proofs are not difficult; rather, the point of the remark is to indicate the extreme power afforded us by intersecting a transformational language with a regular language. Note, further, that the regular languages used are of a rather simple kind. The word 'simple' here can be given a precise meaning in terms of the

number of nestings of cycles generated by a regular grammar. In this sense, the grammars for the regular languages that we have been using are the simplest of all regular languages.

We turn now to the main result of this paper. There is good evidence, as indicated above, that we need to intersect the strings generated by a transformational model of the grammar of a natural language with a regular language in order to produce exactly the grammatical sentences of the natural language. We should therefore want to find a characterization of the class of languages generated by the intersection of a transformational and a regular language. The answer is that we can generate essentially any recursively enumerable set in such a manner.

THEOREM 1: There exists a fixed regular language H such that for any recursively enumerable set of natural numbers S there exists a transformational grammar G such that $\underline{n} \in S$ iff $a^n t \in L(G) \cap H$.[5]

The proof relies on a lemma that is due first to Leonard Haines.

LEMMA 4 (Haines): There exists a fixed context-free language L_1 and a fixed homomorphism λ such that for any recursively enumerable set of numbers S there exists a context-free language L_2 such that $\lambda(L_1 \cap L_2) = \{a^n / n \in S\}$.

The proof of this lemma is forthcoming by Haines, and we shall not repeat it here. We must describe, however, the construction of L_1 and λ for the proof of the theorem.

Let K be the language over the terminal alphabet $\{c, \ldots g\}$ consisting of all those strings which reduce to ϵ (if x reduces to ϵ we shall write R(x)). A string x over this alphabet reduces to ϵ iff it meets one of the following conditions:

(i) $x = \epsilon$

(ii) $(\exists i)(x = cd^i c\ y\ ce^i c)$ & R(y)

(iii) $(\exists i)(x = cf^i c\ y\ cg^i c)$ & R(y)

(iv) $(x = y\ z)$ & R(y) & R(z).

It is clear that we can generate K by some context-free grammar.

Then L_1 is the language over the terminal alphabet $\{a, b, c, \ldots g\}$ as follows: $L_1 = (K \cup \{a, b\}^*)^*$.

λ is defined as follows: $\lambda(a) = a$;

$\lambda(b, \ldots g) = e$, where $\underline{e}$ is the null string.

L_2 is some CF language over the alphabet $\{a, \ldots g\}$.

The outline of the proof is as follows: In the base P of the grammar G we generate two strings side by side, one each from L_1 and L_2, and in addition an adjoining string from the alphabet $\{a, \ldots g\}$. Let us call these strings x, y, z. The transformations then essentially check, one symbol at a time, that x = y = z, and progressively delete x and y at the same time. An additional transformation in each cycle performs the operation of λ on the string z. There is a postcyclic filter that ensures that z is not merely a

proper initial part of the reflections of x and y. If some transformation fails to apply, the string is filtered out. All deletions are in accordance with the principle of recoverability, since we delete items from x and y only as they are equal to something in z, and we delete symbols in z in performing λ by virtue of equality with one of a finite number of terminal strings.[6] A trick is used to ensure that P provides us with the number of S recursions needed to completely refine any string by means of the transformations.

We shall proceed with the details of the description of G. $G = (P, (T_1, T_2, T_3))$, where P is the context-free base defined by the following rules.

$$S \longrightarrow Sa\beta_1 \qquad S \longrightarrow Sa\beta_1\beta_3 t \qquad S \longrightarrow S_1S_2a\beta_1$$
$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$
$$S \longrightarrow Sg\beta_1 \qquad S \longrightarrow Sg\beta_1\beta_3 t \qquad S \longrightarrow S_1S_2g\beta_1$$

$$S \longrightarrow S_1S_2a\beta_1\beta_3 t$$
$$\vdots$$
$$S \longrightarrow S_1S_2g\beta_1\beta_3 t$$

This system of 28 rules can be collapsed into one rule by the well-known convention as follows:

$$S \longrightarrow \left\{ \begin{array}{l} S\left\{\begin{array}{c} a \\ \vdots \\ g \end{array}\right\}\beta_1\ (\beta_3 t) \\ S_1S_2\left\{\begin{array}{c} a \\ \vdots \\ g \end{array}\right\}\beta_1\ (\beta_3 t) \end{array} \right\}$$

S_1 is the initial symbol of the CF grammar that generates language L_1 described above, and S_2 is the initial symbol of the CF grammar that generates language L_2, where L_2 varies according to the recursively enumerable set S.

We shall not write the transformations in the formal notation of Peters and Ritchie, since this notation is somewhat opaque. Rather, we shall use an informal notation close to that which appears in most monographs in linguistics. It can be checked that the operations described informally below do not exceed the capacity of the Peters-Ritchie transformations.

T_1.

Analysis: $[\ x\ ]_{S_1}\ [\ y\ ]_{S_2}\ Z\ \beta_3\ t$

such that (i) x and y are single terminal symbols

(ii) Z is any string over the alphabet.

Operation: delete 'β_3'.

T_2.

Analysis: $[\ X\ x\ ]_{S_1}\ [\ Y\ y\ ]_{S_2}\ Z\ z\ \beta_1\ W$

such that (i) X, Y, Z, and W are any strings, including the null string

(ii) x, y, and z are single terminal symbols

(iii) x = y = z.

Operation: delete x, y, and 'β_1'.

T_3.

Analysis: Z z (t)

such that (i) Z is any string, including the null string

(ii) z is a single terminal symbol from the set $\{b, \ldots g\}$

Operation: delete z.

(Note that T_3 is written as representing really two transformations, one of which operates in the environment 'Z z t', and the other of which operates in the environment 'Z z'.)

Explanation: The language generated by P consists of strings of the form $[_S \ldots [_S [_{S_1}\ x\ ]_{S_1}\ [_{S_2}\ y\ ]_{S_2}\ u_1\beta_1(\beta_3 t)\ ]_S \ldots\ u_n\beta_1(\beta_3 t)\ ]_S$, where $x \in L_1$, $y \in L_2$, and $(u_1 \ldots u_n)$ is an arbitrary string over the terminal alphabet. T_1 operates as a postcyclic filter because of construction of the regular language H below. T_2 checks that the rightmost symbol of 'x' equals the rightmost symbol of 'y' equals $\underline{u}_i$ in each cycle. If this equality holds, the former two and the symbol 'β_1' is deleted. If the equality fails to hold, 'β_1' is left in the interior of the string and it is filtered out by construction of H. T_3 performs the operation of the homomorphism λ in each cycle; it is important that T_3 follow T_2.

Let H consist of all strings over the terminal alphabet of the form $a^n t$, all $\underline{n}$. In an accepted string, '$\beta_3 t$' is generated nowhere in the string except in the first cycle. If it appears in the middle somewhere, or not at all, the string is filtered by H. The string is accepted if $x = y = (u_n \ldots u_1)$. T_2 progressively deletes 'x' and 'y', and T_3 performs $\lambda(u_1 \ldots u_n)$. The following cases exhaustively treat strings generated by P which fail to be in $L(G) \cap H$.

Case 1. $lh(x) < n$, or $lh(y) < n$. Then, is some cycle the SC of T_2 will not be fulfilled, T_2 will fail to apply, 'β_1' will not be erased, and the resulting string will not be in H.

Case 2. $lh(x) = lh(y) = n$, but the three-termed equality fails at some cycle. Then T_2 fails to apply, 'β_1' is not erased, and the resulting string is not in H.

Case 3. $lh(x) > n$ or $lh(y) > n$. Then T_1 will sense on the last cycle that there is more than one symbol left in either 'x' or 'y', it will fail to erase 'β_3', and the resulting string will not be in H.

We have, thus, that for all x, $x \in \lambda(L_1 \cap L_2)$ iff $x \in L(G) \cap H$. (For ease of reading, 'β_1' should be written 'β_2', and 'β_3' written 'β_1'. The incongruity is due to a

modification in ordering of the transformations while the proof was being written.)

Consider now the class of CF-based TG's with arbitrary deletions. A member of this class can be considered as a finite calculating procedure, and thus, by Church's thesis, the class of languages generated by this class of grammars is no greater than the set of all recursively enumerable languages. From these considerations, we then have the following result.

COROLLARY 1: For every CF-based TG with arbitrary deletions, G_1, there exists a regular language H and a CF-based TG which meets the condition on recoverability of deletions such that $L(G_1) = L(G_2) \cap H$.

The answer to the converse question is given in the following theorem.

THEOREM 2: For every recursively enumerable set of natural numbers, S, there exists a CF-based TG with arbitrary deletions such that for all $\underline{n}$, $n \in S$ iff $a^n t \in L(G)$.

Proof: The proof proceeds by imitating the proof of Theorem 1 with a slight modification. That is, we add to the grammar described in the proof above a new transformation, T_4, which is ordered to follow the other transformations. T_4 will delete all terms of its analysis if it ever senses 'β_1' or 'β_3', or if it senses 't' in the interior of a string. Actually, as transformations are now set up, it may be impossible to tell a transformation, "do such-and-such if you sense a certain symbol anywhere besides on the rightmost side of the S-phase in which you are working." If 't' is generated by the base of G somewhere in the interior of a string, however, and 'β_3' is in fact erased by T_1 because the latter's SC is satisfied, then on the next cycle 't' will appear either two or else four symbols in from the right-hand side of the S-phase, because of the definition of P. This predictability in the occurrence of 't' allows us to build a mechanism into T_4 to sense 't' if it ever occurs in the interior of a string.

COROLLARY 2: For every CF-based TG_1 which meets the condition on recoverability and every regular language H there exists a CF-based TG_2 over the same alphabet with arbitrary deletions such that $L(G_1) \cap H = L(G_2)$.

The implications of Corollaries 1 and 2 for linguistic theory are as follows: If we increase the power of a transformational model of a grammar of a natural language by allowing intersection with regular languages, as evidently we must in order to handle filters and postcyclic transformations, then we do not reduce the weak generative capacity of the class of grammars proposed as models by restricting them with the condition of recoverability of deletions. The important side of this equivalence is given by Corollary 1; namely, given a language L generated by a CF-based TG with filters and postcyclic transformations and which uses arbitrary deletions, then we can generate L with a CF-based TG with filters and postcyclic transformations which meets the condition on recoverability of deletions.

In terms of weak generative capacity, then, adding the principle of recoverability of deletions to a CF-based TG with filters and postcyclic transformations is ornamental.

It would be incorrect, however, to conclude that we have therefore a mathematical proof that in the present context of transformational analysis the principle of recoverability of deletions is without empirical content, since the linguist is interested in the strong generative capacity of those grammars that he proposes as models for the grammars of natural languages.

Let us return finally to that with which we began, i.e., IP. Corollary 2 suggests a method by which to prove the negative of IP by proving that CF-based TG's with arbitrary deletions are stronger than CF-based TG's that meet the principle of recoverability of deletions. On this basis, one suspects immediately that IP does, in fact, receive a negative solution, although I have not seen a proof that the condition on recoverability of deletions may not be dispensed with where filters and postcyclic transformations are not employed.

Note, however, that a negative solution to IP does not mitigate the force of the remarks above concerning the power that we are adding to a TG by allowing filters and postcyclic transformations. For Theorem 1 could be rephrased by dropping the part about the regular language H and adding that G is a TG with filters and postcyclic transformations. That is, if the conjecture above does in fact hold, namely, if CF-based TG's with arbitrary deletions are stronger than CF-based TG's with recoverable deletions, then we are increasing the generative power of this latter class of objects by adding filters and postcyclic transformations to them.

We should make the final remark that if the IP receives a negative solution, as we have conjectured that it does, then languages generated by the Peters-Ritchie transformational grammars turn out to be strange mathematical objects. For it is known that the other classes of grammars that have received extensive study, namely, regular grammars, context-free grammars, and context-sensitive grammars, do not increase in generative capacity upon intersection with a regular language.

If the IP does, in fact, receive a negative solution, then we have a characterization of the extent to which the addition of the condition of recoverability of deletions restricts the weak generative capacity of CF-based TG's. This is to say, dropping the condition on recoverability will extend the weak generative capacity of CF-based TG's exactly as much as will intersecting the 'recoverable' languages with regular languages. Furthermore, if IP receives a negative solution, then allowing filters and postcyclic transformation extends the generative capacity of CF-based TG's exactly as much as does intersecting the languages generated by these devices with regular languages. A stronger result, which comes from an analysis of the proof of Theorem 1, is that the addition of filters and postcyclic transformation extends the generative capacity of CF-based TG's exactly as much as the generative capacity of these grammars is restricted by adding the condition on recoverability of deletions, providing that IP has a negative solution.

5. Linguistic Implications

It may be plausibly argued that the results obtained concerning the generative power of certain classes of transformational grammars are of only peripheral interest for linguistics. For, what is of interest linguistically is the internal structure of the recursive devices that serve as models for grammars. That is, suppose that we find (somehow) that we do need a class of devices to serve as models of grammars such that any recursively enumerable set can be generated by one of these devices. This does not mean that the problem of constructing grammatical models for natural languages loses interest on the ground that all we shall have shown is that grammars are, after all, finite computational devices, which is always the implicit assumption in any case. The linguistically interesting properties of grammars reside largely in the formal properties of their rule systems, rather than in their generative capacity.

Generative capacity has been of linguistic interest, in that past, only in a negative sense. That is to say, a characterization of the generative capacity of a class of devices has been useful traditionally only in showing that such kinds of devices can not function as models for grammars. Thus, it can be shown that there are nested dependencies obtaining in natural languages which in principle are beyond the expressive capacity of regular grammars, and presumably the same is true of grammars that generate context-free languages, although several proofs of this now in circulation need revision, since Ullian has shown a conjecture of Haines concerning nondoubling languages to be false.

In this light, we may ask whether the results reported here are of any central interest to linguistics. I believe they are in the sense of suggesting certain changes in the theoretical structure of Aspects. The proof of Theorem 1 does not depend upon deep properties concerning the internal structure of TG's, but rather relies on a series of tricks. These tricks are predicated on the fact that a transformation can read a certain terminal symbol generated by the base which triggers its operation. Note also, that we used this device in the proof of Lemma 2, which is linguistically surprising in view of the past discussions concerning whether transformations in natural language ought to be obligatory, optional or a mixture. The current work in restricting the generative capacity of TG's is concerned with strengthening the condition on recoverability of deletions.[7] One very obvious means, however, of restricting generative capacity which is suggested by the theorems above is that of not allowing transformations to be sensitive to the existence of terminal symbols or terminal strings. Formally, this would mean that we eliminate one kind of predicate from availability for use in the SC of a transformation, namely, the third kind mentioned above.

The constraint of not allowing transformations to 'read' lexical strings has been proposed by others. And we find in our results further reasons of a purely mathematical nature for imposing this constraint. The empirical question that is raised, then, is that

of whether a transformational grammar with this constraint is sufficiently powerful to generate all and only the grammatical sentences of a given natural language.

J. P. Kimball

Footnotes and References

1. I am indebted to Dr. Leonard Haines for many discussions on the material which is presented here. In particular, he suggested the use of his theorem, which appears as a lemma, as a tool by means of which to prove the main result of this report. I am, of course, solely responsible for any errors.
2. N. Chomsky, Aspects of the Theory of Syntax (The M.I.T. Press, Cambridge, Mass., 1965).
3. Ibid., p. 136.
4. Ibid., pp. 138-139.
5. A result obtained subsequent to the writing of this report strengthens this result to the extent of showing that the TG in question may have a regular grammar as a base instead of a context-free grammar. That is, the class of CF-based TG's is no stronger than the class of regular-based TG's. The proof is too lengthy to be included here, but will be submitted for publication elsewhere.
6. Recent discussion concerning syntax has suggested that the condition on recoverability of deletions should be strengthened to require that if terminal string 't_1' is deleted by virtue of identity to terminal string 't_2', the identity must be one of deep structure in addition to simple identity of the debracketed strings. We should then want to know if strengthening the condition in this way restricts weak generative capacity. Let us define the deep structure of a terminal string 't' informally to be the lowest node which dominates all of 't' in the string generated by the base. Following this definition, it is easy to show that the proof of our main theorem can be carried through under this stronger condition on recoverability of deletions. We change the base component so that if a terminal symbol is generated by the rule $A \rightarrow X\ b\ Y$ in the base, then instead it will be generated by the two rules $A \rightarrow X\ B\ Y$, $B \rightarrow b$, where B is a new nonterminal symbol. Thus, by definition, every occurrence of the terminal 'b' will have the same deep structure, to wit, 'B'. We can see that the proof of the main result above goes through under the stronger condition on recoverability of deletions by noting that only terminal strings of length one are deleted by any single application of a transformation.
7. A. B. Peters, personal communication, 1966.