RUSSIAN CONJUGATION

Computer Synthesis of Russian Verb Forms

F. H. H. KORTLANDT*

0. The program presented in this paper is a computerized version of Roman Jakobson's article "Russian Conjugation" in *Word* 4 (1948), 155-67 (abbr.: J). Wherever the program and Jakobson's rules yield different results, this is indicated below in square brackets. The language in which the program is stated is ALGOL 60.

A comprehensive comment is added to the program in order to make it accessible to both linguists without an extensive knowledge of programming languages and mathematicians who are not familiar with the Russian verb system. The input of the computer consists of the program and a list of verb stems. The generated forms are the phonemic representations of the infinitive, the preterit (masc.sg., fem.sg., neuter sg., plural), the present (1st sg., 2nd sg., 3rd sg., 1st pl., 2nd pl., 3rd pl.), and the imperative (2nd sg., 2nd pl.). [The inclusive imperative is not generated because it has no special form (J 2.122). The aspect of the verb is not taken into account; therefore, the irregularly stressed form *rad'ilá* is not generated.

There is one major difference between the language described by Jakobson and the contemporary standard language in favour of which I have decided: the unstressed alternant of the present tense suffix *á* is *u* in Jakobson's paper (J 2.121) but *a* according to the modern standard.[1] For the rest I have stuck rather closely to Jakobson's description.]

* The author studied Mathematics and Linguistics at the University of Amsterdam, where he now is a member of the Staff of the Slavic Department. His dissertation *Modelling the Phoneme: New Trends in East European Phonemic Theory* was recently published by Mouton.

[1] Cf. R. I. Avanesov and S. I. Ožegov, *Russkoe literaturnoe proiznošenie i udarenie* (Moscow, 1960), p. 698: "Proiznošenie ètix form s glasnym [u] v okončanii *-at* (*-jat*), svojstvennoe russkomu literaturnomu jazyku v prošlom, teper' uderživaetsja po preimuščestvu v reči staršego pokolenija, a takže prinjato v sceničeskoj reči"; M. V. Panov, *Russkaja fonetika* (Moscow, 1967), p. 320: "V nastojaščee vremja nado rekomendovat' proiznošenie *to*[p'ăt], *lju*[b'ăt], *lo*[v'ăt] i pr., no staraja orfoèpičeskaja norma ostaetsja dopustimoj."

```
begin comment This program generates the infinitive and the finite inflected forms of Russian verbs;
      integer a,b,v,g,d,e,Z,z,i,j,k,l,m,n,o,p,r,s,t,u,f,x,c,C,S,y,A,E,I,O,U,h,w,
      li,si,fi,di,syl,final,finalcons;
      integer array lexeme[1:20], stem[-2:20], form[-2:25], des[1:2];
      boolean reflexive, drop suffix, simple desinence, vocalic desinence, open, jvnm, soft stem, soften,
      accented, removable accent, final syllable stressed;
      boolean procedure vowel(q); integer q;
      vowel:= q=a ∨ q=e ∨ q=i ∨ q=o ∨ q=u ∨ q=A ∨ q=E ∨ q=I ∨ q=O ∨ q=U;
      boolean procedure soft(q); integer q;
      soft:= q=j ∨ q=C ∨ q=S ∨ q=Z ∨ q=y;
      boolean procedure labial(q); integer q;
      labial:= q=p ∨ q=b ∨ q=f ∨ q=v ∨ q=m;

      procedure omit; fi:=fi-1;
      procedure change(w); integer w; form[fi]:=w;
      procedure add(w); integer w; begin fi:=fi+1; form[fi]:=w end;
      procedure insert(w); integer w; begin fi:=fi+1; form[fi]:=form[fi-1]; form[fi-1]:=w end;
      procedure omits; si:=si-1;
      procedure changes(w); integer w; stem[si]:=w;
      procedure adds(w); integer w; begin si:=si+1; stem[si]:=w end;
      procedure inserts(w); integer w; begin si:=si+1; stem[si]:=stem[si-1]; stem[si-1]:=w end;
      procedure add desinence; begin integer h; for h:=1 step 1 until di do add(des[h]) end;
      procedure transfer; begin integer h; for h:=1 step 1 until si do form[h]:=stem[h]; fi:=si end;
      procedure stress(w); integer w; w:=w+27;
      procedure unstress(w); integer w; if w>36 then w:=w-27;

      procedure stress last vowel;
      begin integer h; boolean unstressed; unstressed:=true; for h:=fi step -1 until 1 do if unstressed ∧ vowel(form[h]) then
            begin stress(form[h]); unstressed:=false
            end
      end;

      procedure accent;
      begin integer h; if removable accent ∧ vocalic desinence then
            begin for h:=1 step 1 until fi do if vowel(form[h]) then unstress(form[h]); stress(des[1]); add desinence
            end else if ¬accented then
            begin if open∧syl>1 ∨ ((final=n ∨ final=m)∧syl=0 ∨ (open ∨ jvnm)∧¬des[2]=a) ∧ ¬vocalic desinence then
                  begin if simple desinence ∧ vowel(des[1]) then stress(des[1]) else stress last vowel; add desinence
                  end else
                  begin add desinence; stress last vowel
                  end
            end else add desinence
      end accent;
```

1. THE PROGRAM

The program consists of a number of declarations, which serve to define certain properties of the variables and procedures used in the program and to associate them with identifiers, and a number of statements (units of operation), which begin in line 65 (label: CONSTANT VALUES). There are several kinds of statements. An assignment statement (e.g., a := 10) serves for assigning a value to a variable or procedure identifier. A go to statement (e.g., *goto* INPUT) interrupts the normal sequence of operations: the next statement to be executed will be the one having the indicated label. A conditional statement (e.g., *if* w = 101 *then* EXIT *else* ...) causes a statement to be executed or skipped depending on the value of a boolean expression. (The value of a boolean expression is either *true* or *false*.) A for clause (e.g., *for* h := 1 *step* 1 *until* 20 *do*) causes the statement which it precedes to be repeatedly executed (i.c. for h = 1, 2, ..., 20). A procedure statement (e.g., phonemics) serves to invoke the execution of a procedure body. This procedure body can be found in the corresponding procedure declaration (i.c. the lines 41-55 of the program), which contains a number of statements that may be preceded by a number of declarations of local identifiers (i.c.: h, q, no vowel, unvoice).[2]

2. COMMENT ON THE PROGRAM

The numbers refer to the respective lines of the program.

2. *a*, ..., *S* represent the phoneme inventory of Russian in the order of the Cyrillic alphabet (*a*, ..., *š*); *y* represents the feature of softness of the preceding consonant ('); *A*, ..., *U* represent the vowels under stress (*á*, ..., *ú*); *h* and *w* are indices.

3. *li, si, fi, di* are indices that indicate the length of the relevant part of the array *lexeme, stem, form, des* respectively; *syl* is the number of syllables of the input stem; *final* and *finalcons* are the final symbol (= phoneme or the feature of softness) resp. the final consonant (or the feature of softness) of the input stem.

4. The array *lexeme* contains the input stem and remains intact during the execution of the program until a new input stem is read.

[2] Cf. *Revised Report on the Algorithmic Language ALGOL* 60, ed. by Peter Naur (Copenhagen, 1964) (corrected reprint).

Changes that have bearing on a whole tense (e.g., *p'isát'*, *p'išú*) are carried out in the array *stem*, while those affecting only one form (e.g., *l'ub'it'*, *l'ubl'ú*) are carried out in the array *form*. The array *des* contains the desinence until it is added to the array *form*. The declaration of a separate array *des* is necessary in view of the accent pattern of the Russian verb, cf. below (procedure *accent*). The suffix of reflexivity (if required) is added in the output procedure. [In Jakobson's paper reflexive verbs are not taken into account.] The numbers in square brackets indicate the lower and upper bound of the subscripts.

5, 6. The booleans *reflexive* and *drop suffix* indicate the reflexivity of the verb and the dropping of the suffix *nu* in preterit forms respectively, see line 72 below. The other booleans correspond to Jakobson's categories, cf. the assignment statements in 78-83 below; *jvnm* stands for Jakobson's 'narrowly closed', *soften* reflects the condition for softening of *hard* consonants. [The categories of simple and vocalic desinence affect the resulting verb forms only through the procedure *accent*. Since this procedure is not used in the generation of the infinitive (84-106 below), these boolean variables are not assigned their proper, 'Jakobsonian' logical values during the execution of this part of the program: they acquire their preterit value (*false*) in line 78 already. Of course, this is only a matter of programming economy and has nothing to do with the linguistic facts.]

7-12. The meaning of these boolean procedures is self-evident: q is a vowel if $q = a$ or $q = e$ or ..., and otherwise it is not, etc. (The symbols $\wedge$, $\vee$, $\neg$ mean *and*, *or*, *not* respectively.) In this program, as in Jakobson's paper (fn. 5 on p. 157), *soft* has the meaning of 'palatal or palatalized'. [There is one slight difference: the phonemically relevant softness of non-palatal sounds is here expressed by the separate symbol *y*, so strictly speaking only this feature is *soft* in the sense of the program while the preceding consonant is not, cf. line 10.]

13. The procedure *omit* reduces the length of the relevant part of the array *form* by one, so that its effect is the omission of the last relevant symbol of this array.

14. The procedure *change* assigns to the last relevant symbol of the array *form* the value of the indicated symbol, so in effect it changes the (value of the) last relevant symbol of the array *form*. This procedure is equivalent to *omit* plus *add*.

15. The procedure *add* lengthens the relevant part of the array *form* by one and fills this place with the (value of the) indicated symbol. So in effect it adds the indicated symbol to the relevant part of the array *form*.

16. The procedure *insert* lengthens the relevant part of the array *form* by one and inserts the indicated symbol before the last relevant symbol of the array *form*. It is equivalent to *omit* plus *add* (the indicated symbol) plus *add* (the omitted symbol).

17-20. These procedures are identical with the ones in 13-16 but for the array that is affected: here the array *stem* is altered, instead of the array *form*.

21. The procedure *add desinence* adds the relevant part of the array *des* to the relevant part of the array *form*.

22. The procedure *transfer* makes the array *form* equal to the array *stem* as far as the elements of the array are relevant.

23. The procedure *stress* changes *a* into *A* etc., see 65, 66 below (CONSTANT VALUES).

24. The procedure *unstress* changes *A* into *a* etc. The if clause is necessary because of the for statement in line 32: if the desinence is stressed, all vowels of the stem are unstressed. (Alternative solutions for this little problem can easily be found and some of them are quicker but they all yield more complicated boolean expressions. Since the number of vowels in the input stem hardly ever exceeds three, I have chosen here for simplicity of the program rather than for rapidity in the execution.)

25-29. This procedure has an obvious meaning. A local boolean is introduced for the sake of clarity, but it can be dispensed with if use is made of a go to statement:

for h := fi *step* −1 *until* 1 *do if* vowel(form[h]) *then*
begin stress(form[h]); *goto* LABEL
end; LABEL:

30. The procedure *accent* adds an accent to unstressed verb forms. Within this procedure the desinence is added to the array *form*.

31, 32. If the desinence begins with a vowel the stress of verbs having a removable accent is on this vowel (J 2.61).

33-37. For verbs with a mobile accent (unaccented input stem) the following rules hold (J 2.62):

(a) verbs with open polysyllabic input stems stress either the simple desinence, or the preceding vowel if the desinence is complex: *p'isa-* 'write', *p'išú, p'íšut, p'iší*;

(b) if the desinence does not begin with a vowel verbs with an input stem ending in *j, v, n, m* or a vowel [= all but the BROADLY closed input stems] stress the vowel that precedes the desinence, with the limitation that in the fem.sg. preterit (suffix *la*) only verbs with nonsyllabic input stems ending in a nasal follow this rule (J 2.24): *žda-* 'wait', *ždalá, ždál'i,*

```
procedure phonemics;
begin integer h,q; boolean no vowel;
        procedure unvoice(w); integer w;
        if w=g then w:=k else if w=b then w:=p else if w=v then w:=f else
        if w=d then w:=t else if w=z then w:=s else if w=Z then w:=S;
        no vowel:= true; for h:=1 step 1 until fi do if vowel(form[h]) then no vowel:= false; if no vowel then insert(O);
        if (form[fi]=j ∨ form[fi]=C)∧form[fi-1]=O∧soft(form[fi-2]) then form[fi-1]:=E;
        for h:=1 step 1 until fi do if soft(form[h-1]) then
        begin if form[h]=e ∨ form[h]=o ∨ form[h]=a∧fi-h>1 then form[h]:=i
        end else if form[h]=o then form[h]:=a else if form[h]=y ∧ (form[h+1]=r ∨ form[h+1]=j) then
        begin for q:=h step 1 until fi-1 do form[q]:=form[q+1]; omit
        end; if form[fi]=y then
        begin unvoice(form[fi-1]); if form[fi-1]=k then omit
        end else unvoice(form[fi])
end phonemics;

procedure output;
begin integer h; for h:=-2 step 1 until fi do
        begin if form[h]>36∧form[h]<63 then
                begin if vowel(form[h]) then PUSYM(126) else PUSYM(127); PUSYM(form[h]-27)
                end else PUSYM(form[h])
        end; if reflexive then
        begin if form[fi]=t then PUSYM(c) else PUSYM(s); if vowel(form[fi]) then PUSYM(y) else PUSYM(a)
        end reflexive
end output;


CONSTANT VALUES: a:=10; b:=11; c:=12; d:=13; e:=14; f:=15; g:=16; i:=18; j:=19; k:=20; l:=21; m:=22; n:=23; o:=24; p:=25;
r:=27; s:=28; t:=29; u:=30; v:=31; x:=33; z:=35; A:=37; E:=41; I:=45; O:=51; U:=57; C:=39; S:=55; Z:=62; y:=120;
BEGIN: li:=si:=fi:=di:=syl:=0; for h:=1 step 1 until 20 do lexeme[h]:=0; for h:=-2 step 1 until 8 do
stem[h]:=form[h]:=93; RUNOUT; PUNLCR; PUNLCR; PUSYM(74); PUSPACE(2); reflexive:=drop suffix:=false;
INPUT: w:=RESYM; if w=101 then EXIT else if w=119 then goto INPUT else PUSYM(w); if w=65 then
begin for h:=1 step 1 until li do stem[h]:=lexeme[h]; si:=li
end else
begin if w=121 then reflexive:=true else if w=98 then drop suffix:=true else if ¬w=99 then
        begin li:=li+1; if li>20 then goto BEGIN else if w=126 ∨ w=127 then
                begin lexeme[li]:=27; li:=li-1
                end else lexeme[li]:=lexeme[li]+w
        end; goto INPUT
end input;
final:=lexeme[li]; simple desinence:= vocalic desinence:= false; open:= vowel(final);
jvnm:= final=j ∨ final=v ∨ final=n ∨ final=m; finalcons:= if open then lexeme[li-1] else final;
soft stem:= soft(finalcons); accented:= final syllable stressed:= false; for h:=1 step 1 until li do if vowel(lexeme[h]) then
begin syl:=syl+1; if lexeme[h]>36 then accented:= final syllable stressed:= true else final syllable stressed:= false
end; removable accent:= ¬jvnm ∧ final syllable stressed;
soften:= ¬soft stem ∧ syl>1 ∧ (final=a∨final=o∨final=A∨final=O);
```

etc., but *žm-* 'press' and *žn-* 'reap' both *žála*;

(c) otherwise the last vowel of the verb form is stressed: *v'od-* 'conduct', *v'idú, v'idút, v'ól, v'ilá, v'ilí.*

N.B.: Monosyllabic verb forms are neither stressed nor unstressed for the presence of either of these characteristics requires the presence of the other elsewhere in the word.[3] The program, however, necessarily distinguishes between stressed and unstressed vowels, so a choice is to be made. Since the vowel oppositions that exist in monosyllabic words are the same as can be found in the stressed syllables of polysyllabic words, I have chosen to consider the vowels of monosyllabic verb forms as stressed. (The same choice was made by Jakobson.) Therefore, a 'stressed' vowel is the stressed vowel of a polysyllabic verb form or the only vowel of a monosyllabic verb form.

39. In other cases the accent is not changed.

41. The procedure *phonemics* contains the morphophonemic rules of the Russian verb.

43-45. The procedure *unvoice* is declared within the procedure *phonemics* and therefore has a local character. Its meaning is obvious.

46. If a verb form contains no vowel, an *ó* is inserted before its final consonant (J 2.5), e.g., *žg-* 'burn', *žók, žglá*; *id-* 'go', *šól, šlá.* Instead of a local boolean (*no vowel*) a go to statement could have been used: *if* vowel(form[h]) *then goto* LABEL; insert(O); LABEL:

47. This *ó*, like any other *ó*, is replaced by *é* if it is preceded by a soft consonant and followed by word-final *j* or *č*: *p'j-* 'drink', *p'éj*; *p'ok-* 'bake', *p'éč.* [Jakobson's rule on inserted vowels runs as follows (J 2.5): "A vowel is inserted within a nonsyllabic full-stem before a nonsyllabic desinence and, if this stem ends in *r*, before any consonantal desinence. The inserted vowel is *é* in the Infinitive, *ó* elsewhere." According to this rule, the imperative of *p'j-* is **p'ój* instead of *p'éj.* A rule changing both this *ó* and the *ó* of **p'óč* into *é* is lacking in Jakobson's description, though the necessity of having such a rule is explicitly referred to in the introduction (J 1.32): "*o* is not admitted between two soft consonants of the stem". However, this formulation of the rule is incorrect because there exist such forms as *l'óža* 'lying' and *jóžitca* 'to cower'.]

48, 49. If preceded by a soft consonant, unstressed *e* and *o* are reduced to *i.* The same holds true for unstressed *a* if it is not the final or prefinal phoneme of a word form, e.g., *tr'as-* 'shake', *tr'isút*, but *v'íd'e-* 'see', *v'íd'at.* [No vowel reduction rules are explicitly stated in Jakobson's

[3] See C. L. Ebeling, "On Accent in Dutch and the Phoneme /ə/", *Lingua* 21 (1968), 135f. on the essence of configurational features.

paper.]

50, 51. Elsewhere (i.e. word-initially and after a hard consonant) unstressed *o* is reduced to *a*. The feature of softness *y* is omitted before *r*, where the softness is lost, and before *j*, where it is automatic. [Jakobson maintains the soft sign ' before *j*. So did I only in the input stem. In fact, it could have been omitted there as well for it does not convey any information at all. If it had actually been omitted, it would have to be inserted before the vowel insertion in the infinitive (and preterit) and the imperative (*pj-/p'j-* 'drink', *p'ít'*, *p'éj*). Since any decision on the input stem is more or less arbitrary, for once I have preferred to stick to Jakobson's notation.]

52-54. Word-final consonants are unvoiced; in addition, word-final velars lose their softness: *l'og-* 'lie down', imperative *l'ák*. [This rule is not explicitly stated in Jakobson's paper.]

56. The procedure *output* punches the array *form* into the output tape and adds the required suffix to reflexive verb forms.

57. Every verb form is preceded by three spaces, cf. the assignment statement in line 68.

58-60. Within the program capital letters (which have values between 36 and 63, see line 66 of the program) stand for stressed vowels and palatal consonants (see line 2 of the program). However, in the output tape (and also in the input tape, see below) stressed vowels are marked by underlining and the háček on a palatal consonant is shown as a vertical line through the letter. The punching of either of these symbols precedes the punching of the letter which is marked by it (like the printing of an accent on an ordinary typewriter). So two symbols are punched instead of a capital letter: the 'accent' is punched first (symbol no. 126 which is underlining if it is a vowel, otherwise symbol no. 127 which is the vertical line) and followed by the corresponding non-capital letter, the value of which is 27 less than the value of the capital letter (cf. the procedures *stress* and *unstress* above). The procedure PUSYM punches the symbol with the indicated value into the output tape.

61, 62. The suffix of reflexivity is *ca* after *t*, *sa* after any other consonant, *s'* after a vowel: *sm'ejá"*- 'laugh', *sm'ijótca*, *sm'ijálsa*, *sm'ijálas'*.

65, 66. Here the statements of the program begin. The letters used in the program are assigned constant values. These values are equal to the values that correspond with the letters in the procedures RESYM and PUSYM, so *a* is symbol no. 10, etc. There is one exception: symbol no. 120 is ' instead of *y*, so an apostrophe in the input tape is read as *y* and

PUSYM (*y*) punches an apostrophe into the output tape. This concession to ALGOL is necessary because an apostrophe cannot be an identifier and therefore such expressions as *add*(') are senseless.

67, 68. This is where the execution of the program really gets started. The indices that indicate the lengths of the relevant parts of the arrays and the number of syllables of the input stem are set equal to zero. The elements of the array *lexeme* are assigned the value 0 because of the assignment statement in line 75. The elements of the arrays *stem* and *form* must have some value as soon as they may appear in an if clause (e.g., in 140-60). For the sake of convenience and in order to have every verb form preceded by three spaces (see line 57), all these elements are assigned the value of a space, which is 93. RUNOUT punches 20 cm of blank tape. PUNLCR punches the value of a new line and carriage return. When the input stem is punched into the output tape, it is preceded by symbol no. 74, i.e. >, and two spaces. (PUSPACE(2) is equivalent to PUSYM(93); PUSYM(93).)

69, 70. The procedure RESYM reads a symbol of the input tape and *w* is assigned the value of this symbol. Symbol no. 101, i.e.], marks the end of the input. When this symbol is read the procedure EXIT makes an end to the execution of the program. The new line and carriage return symbol, which has value 119, is skipped and the execution starts once more from the label INPUT at the beginning of this line, so the next symbol of the input tape is read. Any other symbol (including a space) is punched into the output tape. Symbol no. 65, which is a hyphen, marks the end of the input stem. When this symbol is read, the array *stem* is made equal to the array *lexeme*, which then contains the letters of the input stem; otherwise, the lines 72-75 are executed and after line 76 the execution starts again from the label INPUT, that is, the next symbol of the input stem is read.

72. The reflexivity of a verb is marked by the presence of the symbol ", which has value 121, in the input stem, e.g., *sm'ejá*"- (or "*sm'ejá*-) 'laugh'. When this symbol is read, the boolean *reflexive* is assigned the value *true*. Otherwise, the value is *false* because of the assignment statement in line 68. (A boolean variable must have one of these two values as soon as it appears in an if clause.) The suffix *nu* is in brackets if it is dropped in the preterit (cf. J 2.23), e.g., *gás(nu)*- 'be extinguished'. The opening bracket (= symbol no. 98) causes the boolean *drop suffix* to be assigned the value *true*, the closing bracket (= symbol no. 99) is skipped.

73-75. Any symbol which has not been mentioned thus far is now added to the array *lexeme*, so the index *li* is increased by one. If *li*

```
INFINITIVE: if jvnm then
begin omits; if final=j then
        begin if syl=0 then adds(i) else if syl=1 then
                begin if stem[si]=o then begin changes(y); adds(e) end else if stem[si]=0 then changes(I)
                end
        end else if (final=n ∨ final=m) ∧ syl=0 then adds(a)
end truncate jvnm; transfer;
if final=k ∨ final=g then
begin change(C); if ¬accented then stress last vowel
end else
begin if final=t ∨ final=d ∨ final=z ∨ final=b then
        begin if form[fi-1]=s then omit else change(s)
        end else if syl=0∧final=r then
        begin insert(e); add(y); add(e)
        end; add(t); add(y); if ¬accented then
        begin if form[fi-2]=s then add(I) else stress last vowel;
                deviating forms: if form[1]=i∧form[2]=s∧fi=5 then
                begin for h:=2 step 1 until 4 do form[h]:=form[h+1]; omit
                end else if form[1]=k∧form[2]=i∧form[4]=A∧fi=6 then
                begin omit; insert(s); add(y)
                end deviating forms
        end
end; phonemics; PUNLCR; if reflexive∧form[fi]=y then omit; output;


PRETERIT: if final=t ∨ final=d then omits else if drop suffix then si:=si-2 else if syl=0∧final=r then
begin inserts(0); accented:=removable accent:=true
end; if si=1 then changes(S); transfer;
if vowel(form[fi]) ∨ si=1 then add(1); if ¬accented then stress last vowel; phonemics; PUSPACE(3); output;
tolok: if stem[1]=t∧stem[2]=o∧stem[3]=l∧stem[5]=k∧si=5 then
begin lexeme[4]:=stem[4]:=k; li:=si:=4
end tolok; transfer;
di:=2; des[1]:=l; des[2]:=a; pryad: if form[1]=p∧form[2]=r∧form[4]=A∧fi=4 then
begin form[4]:=a; des[2]:=A
end pryad; if final=j ∧ ((form[1]=b∨form[1]=S)∧syl=0 ∨ form[fi]=e∧syl=1) then
begin stress last vowel; add desinence
end else accent; phonemics; output;
transfer; des[2]:=o; accent; phonemics; output;
if form[fi]=0 then begin change(y); add(I) end else begin change(y); add(i) end; output;
```

becomes too large (cf. the declaration of the array *lexeme* in line 4), the execution starts again from the label BEGIN as if line 200 had been reached. Otherwise, *lexeme*[*li*] is assigned the value of the symbol that has been read. There is one complication: stressed vowels and palatal consonants are compressed into one symbol, which is the corresponding capital letter. This means that when the stress mark of a vowel (symbol no. 126) or the háček of a consonant (symbol no. 127) is read, the symbol to be read next is to be changed into a capital letter, i.e. to be increased by 27 (cf. the comment on 58-60). This compression of stressed vowels and palatal consonants is not really necessary, but it induces considerable simplifications in a number of statements, e.g. in the procedures *stress* and *unstress*, the softening rules of the present tense, and the condition for the desinence of the imperative (line 197). The palatalized consonants could similarly be compressed, but here the need to do so is less urgent and it would necessitate a number of changes in various statements that would not further the readability of the program.

76. Now the next symbol of the input stem is to be read unless the previous symbol was a hyphen.

78-82. The meaning of these assignment statements will be obvious to anyone who has succeeded in reading the program so far. The values of the booleans *accented* and *final syllable stressed* are set equal to *false* before the for statement that counts the syllables (the number of syllables is equal to the number of vowels) and in which the values of these booleans are set equal to *true* when a stressed vowel occurs. Then *final syllable stressed* becomes *false* again if there follows an unstressed vowel. Open and broadly closed input stems have a removable accent if their final syllable is stressed (J 2.61).

83. The final consonant of hard polysyllabic input stems ending in *a* or *o* is softened in the present and the imperative (J 2.42.A), see 161-69.

84-90. These lines contain the changes of the stem that are common to the infinitive and the preterit forms. Input stems in *j*, *v*, *n*, *m* drop their terminal phoneme (J 2.221). Nonsyllabic stems in *j* replace this *j* by *i*, e.g., *p'j-* 'drink', *p'ít'*, while monosyllabic stems replace the vowel *o* before the dropped *j* by *i* [if stressed, otherwise by *e* which (unlike *i*) palatalizes the preceding consonant], e.g., *mój-* 'wash', *mít'* (cf. *mójit*) [but *poj-* 'sing', *p'ét'* (cf. *pajót*)]; nonsyllabic stems in *n*, *m* replace this nasal by *a*, so *žm-* 'press' and *žn-* 'reap', both *žát'* (J 2.24). After these changes the stem is transferred to the array *form*, where the infinitive is generated. The array *stem* is preserved for the formation of the preterit in 107ff.

91, 92. Verbs with an input stem ending in a velar have an infinitive ending in *č* (J 2.112 and 2.222), e.g., *p'ok-* 'bake', *p'éč*.

94, 95. All terminal dentals and labials of the broadly closed input stems coalesce into *s* before the infinitive desinence (J 2.3), e.g., *gr'ob-* 'row', *gr'ist'í*. [This *s* coalesces with a preceding *s* into one phoneme: *rost-* 'grow', *rast'í*.]

96, 97. The infinitive of nonsyllabic stems in *r* shows *polnoglasie* (J 3.1), e.g., *t'r-* 'rub', *t'ir'ét'*.

98, 99. The infinitive desinence is *t'í* in verbs with unaccented input stems if the desinence is preceded by a consonant [which must of course be *s*], *t'* elsewhere (J 2.112). The desinence is here directly added to the array *form*.

100-104. There are two exceptions (J 3.1): *id-* 'go', *it'í* instead of **ist'í* and *kl'an-* 'curse', *kl'ást'* instead of **kl'át'*.

106. The infinitive is punched after a new line and carriage return symbol. If it ends in *t'* and the verb is reflexive, the soft sign is omitted before the suffix *ca* (which is added in the output procedure, 61f.), e.g., *sm'ejá"-* 'laugh', *sm'ijátca*.

107, 108. Now we return to the array *stem*. If the input stem ends in *t* or *d* this dental stop is dropped before the preterit desinences (J 2.222). So is the suffix *nu* if it was in brackets in the input stem (J 2.23), cf. line 72. Nonsyllabic input stems in *r* insert a stressed *o* in the preterit forms (J 2.5), e.g., *t'r-* 'rub', *t'ór*, *t'órla*. (The accent is removable, however, because the desinence is stressed in the present and the imperative where there is no inserted vowel.)

109. The stem *id-* 'go', which lost its dental stop in line 107 so that its length has been reduced to one, is replaced by *š* in the preterit: *šól* (vowel insertion in line 46 of the procedure *phonemics* which is invoked in line 110), *šlá*. After these changes, which are common to all preterit forms, the stem is transferred to the array *form* where the desinence and the accent are added (if necessary).

110. The masc.sg. desinence is *l* after a vowel and zero after a consonant (J 2.111). [The form *šól* is not mentioned by Jakobson.] The output is preceded by three extra spaces.

111-13. [The correct preterit forms of the verb *tolok-/tolk-* 'pound', *talóč*, *talók*, *talklá*, *talčót* are not found in Jakobson's paper. The forms of the infinitive and the masc.sg. preterit are derived from one stem and all other forms from the other. The alternation cannot be automatic because there exist such verbs as *volok-* 'drag', *valóč*, *valók*, *valaklá*, *valačót* and *mólk(nu)-* 'become silent', *mólknut'*, *mólk*, *mólkla*, *mólkn'it*.

I have not been able to find a more elegant solution to this problem.] Once more the stem is transferred to the array *form*.

114-18. The fem.sg. desinence is *la* (J 2.111). The form *pr'ilá*, derived from the input stem *pr'ád-* 'spin', is irregularly stressed. [According to Jakobson the input stem is *pr'ad-* and the irregularly stressed forms are *pr'ála*, *pr'ál'i* (J 3.2). But this is incorrect because his input stem would generate the infinitive **pr'ist'í* instead of *pr'ást'*, a fact which he clearly failed to realize.] The forms *b'íla* and *šíla*, derived from the input stems *b'j-* 'beat' and *šj-* 'sew' respectively, are also irregularly stressed [these verbs are not even mentioned by Jakobson], and so is the form *p'éla* from *poj-* 'sing'. [The irregular stress pattern of the latter verb is not stated explicitly though it can be derived from the form that is mentioned in J 3.2 because of its vowel alternation.]

119. The neuter sg. desinence is *lo* (J 2.111). It is necessary to invoke the procedures *transfer*, *accent*, *phonemics* once more because the stress is often different from the stress on the previous form and the proper quality of the unstressed vowels is lost in the procedure *phonemics*.

120. The plural desinence is *l'i* (J 2.111). Since this form is for the rest identical with the previous one, it is unnecessary to invoke the procedures of line 119 again.

121. For the generation of the present and the imperative forms the original input stem is once more transferred to the array *stem*.

122-27. The present of the verbs *daj-* 'give' and *jéd-* 'eat' is so irregular (J 3.3) that a PUTEXT seems to be justified. Any other solution would require a great number of *ad hoc* rules. The procedure PUTEXT punches the indicated string into the output tape.

128. The 1 sg. desinence of all other verbs is *u* (J 2.121). Open input stems lose their final phoneme before the present and imperative desinences (J 2.21).

129-35. The present stem of verbs with an input stem in *ova* ends in *uj* (J 2.24). If the final syllable of the input stem is stressed, the stress moves to the desinence in two-syllabic stems only, otherwise to the preceding vowel, e.g., *ková-* 'forge', *kujú*, but *darová-* 'grant', *darúju*. There is one exception (J 3.2): *dn'ová-* 'spend the day', *dn'úju*. If the final *a* of the input stem is not stressed, the accent remains unchanged.

136, 137. The present stem of verbs with an input stem in *aváj* ends in *aj* and the stress moves to the desinence (J 2.23): *daváj-* 'give', *dajú*. The input stem reappears in the imperative *daváj*, cf. below (line 194).

139. The following verbs show an irregular alternation (J 3.2):[4]

[4] Professor F. E. J. Kruseman Aretz pointed out to me that the execution of the

```
PRESENT: simple desinence:= vocalic desinence:= true; for h:=1 step 1 until li do stem[h]:=lexeme[h]; si:=li;
daj: if stem[1]=d∧stem[2]=a∧stem[3]=j∧si=3 then
begin PUNLCR; PUTEXT(⊰  dam  da$  dast  dad'im  dad'it'i  dadut⊱); goto IMPERATIVE
end daj;
jed: if stem[1]=j∧stem[2]=E∧stem[3]=d∧si=3 then
begin PUNLCR; PUTEXT(⊰  jem  je$  jest  jid'im  jid'it'i  jid'at⊱); stem[3]:=S; removable accent:=false; goto IMPERATIVE
end jed;
di:=1; des[1]:=u; if open then
begin omits; if (final=a∨final=A)∧stem[si]=v∧stem[si-1]=o then
        begin omits; removable accent:= false; if final syllable stressed then
                begin if syl=2∧¬(stem[1]=d∧stem[2]=n) then
                        begin changes(u); stress(des[1])
                        end else changes(U)
                end else changes(u); adds(j)
        end ova
end else if final=j∧stem[si-1]=A∧stem[si-2]=v∧stem[si-3]=a then
begin si:=si-2; changes(j); stress(des[1])
end truncation;

Discrepancy between the prevocalic and the preconsonantal stem shape:
if stem[1]=b∧stem[3]=e∧stem[4]=Z∧si=4 then begin changes(g); finalcons:=g end else
if stem[1]=o∧stem[2]=r∧si=2∧final=A ∨ stem[1]=s∧stem[2]=o∧stem[3]=s∧si=3 ∨
stem[1]=s∧stem[2]=t∧stem[3]=o∧stem[4]=n∧si=4 ∨ stem[1]=Z∧stem[2]=A∧stem[3]=Z∧stem[4]=d∧si=4 then soften:= false else
if stem[1]=s∧stem[2]=p∧si=2 then begin adds(y); finalcons:=y; soft stem:= true end else
if stem[1]=r∧stem[2]=y∧stem[3]=o∧stem[4]=v∧si=5 then begin si:=4; soft stem:= false end else
if stem[1]=s∧stem[2]=m∧stem[4]=e∧stem[5]=j∧si=5 ∨ stem[1]=r∧stem[2]=Z∧si=2 then soft stem:= false else
if stem[1]=r∧stem[2]=o∧stem[3]=p∧si=4 ∨ (stem[1]=k∧stem[2]=l∧stem[4]=e∧stem[5]=v∧stem[7]=e ∨ stem[1]=s∧stem[2]=k∧stem[3]=r∧
stem[5]=e∧stem[6]=Z∧stem[7]=e ∨ stem[1]=t∧stem[2]=r∧stem[4]=e∧stem[5]=p∧stem[7]=e)∧si=8 then inserts(s) else
if stem[1]=s∧stem[2]=l∧si=2 then begin stem[1]:=S; adds(y) end else
if stem[1]=j∧stem[2]=E∧stem[3]=x∧si=3 then begin changes(d); soften:= false end else
if stem[1]=m∧stem[2]=o∧stem[3]=l∧si=3 then begin omits; changes(y); adds(e); adds(l) end else
if stem[1]=b∧stem[2]=r∧stem[4]=I∧stem[5]=j∧si=5 then stem[4]:=E else
if stem[1]=z∧stem[2]=v∧si=2 then inserts(o) else
if (stem[1]=b∨stem[1]=d)∧stem[2]=r∧si=2 then begin inserts(y); inserts(e) end else
if stem[1]=l∧stem[3]=E∧stem[4]=z∧si=4 then removable accent:= false else
multiple discrepancy:
if stem[1]=g∧stem[2]=n∧si=2∧final=a then begin inserts(o); adds(y); syl:=2; soft stem:= true end else
if stem[1]=s∧stem[2]=t∧stem[3]=l∧si=3 then begin inserts(y); inserts(e); adds(y); syl:=2; accented:= removable accent:= false end else
if stem[1]=l∧stem[3]=o∧stem[4]=g∧si=4 then begin omits; changes(A); adds(g); accented:= true end else
if stem[1]=s∧stem[3]=E∧stem[4]=d∧si=4 then begin omits; changes(A); adds(d); removable accent:= false end else
if si=1 then begin adds(U); adds(d); accented:= true end discrepancy;
```

140. *b'ežá-* 'run', *b'igú*;

141, 142. *orá-* 'shout', *arú*; *sosá-* 'suckle', *sasú*; *stona-* 'groan', *stanú*; *žážda-* 'thirst', *žáždu*;

143. *spa-* 'sleep', *spl'ú* (palatalization in line 177);

144. *r'ov'é-* 'roar', *r'ivú*;

145. *sm'ejá"-* 'laugh', *sm'ijús'*; *ržá-* 'neigh', *ržú*;

146, 147. *ropta-* 'grumble', *rapššú*, etc. (palatalization in line 163);

148. *slá-* 'send', *šl'ú*;

149. *jéxa-* 'drive, go', *jédu*;

150. *molo-* 'grind', *m'il'ú* (palatalization in line 167);

151. *br'íj-* 'shave', *br'éju* [here, as in all other verbs which show discrepancy, I have chosen the infinitive stem shape as the representation of the input stem; the vowel alternation of the verb *poj-* 'sing' is generated by the program];

152. *zva-* 'call', *zavú*;

153. *bra-* 'take', *b'irú*; *dra-* 'tear', *d'irú*;

154. *l'éz-* 'climb', *l'ézu*; here the verb *mog-* 'can' must be added, which has the accent pattern (but not the palatalization) of open polysyllabic input stems [this verb is not mentioned by Jakobson];

156. *gna-* 'drive', *gan'ú*;

157. *stlá-* 'spread', *st'il'ú* [the accent of Jakobson's *stlalá*, which requires an unstressed input stem, is confirmed by none of my informants and rejected by Avanesov and Ožegov in their *Russkoe literaturnoe proiznošenie i udarenie* (Moscow, 1960), p. 567];

158. *l'og-* 'lie down', *l'águ*;

159. *s'éd-* 'sit down', *s'ádu*;

160. *bi-* 'be', *búdu* (the *i* of the input stem has already been omitted in line 129).

161. If the condition of line 83 is fulfilled, the final consonant of the stem is softened (J 2.42.A). Examples:

162, 163. *pláka-* 'weep', *pláču*; *skaka-* 'jump', *skačú*; *iska-* 'look for', *iššú*; *pr'áta-* 'hide', *pr'áču*; [I prefer to write *šš* instead of *šč* in accordance with the modern standard pronunciation[5]]

---

program takes considerably less time if special variables are introduced to be used in the discrepancy rules instead of the subscripted variables of the array *stem* (e.g., s1 := stem [1], etc.) because the looking up of a subscripted variable takes relatively much time.

[5] Cf. Avanesov and Ožegov, 1960:682: "Na meste bukvy *šč* proiznositsja dvojnoj mjagkij soglasnyj [š'š']"; Panov, 1967:331: "Poètomu bezuslovno nado sčitat' rekomenduemoj normoj proiznošenie [šš']."

```
        if soften ∧ ¬stem[si]=j then softening:
        begin if stem[si]=k ∧ ¬stem[si-1]=t ∨ stem[si]=t then
                begin if stem[si-1]=s then begin omits; changes(S); adds(S) end else changes(C)
                end else if stem[si]=g ∨ stem[si]=d ∨ stem[si]=z then
                begin omits; if stem[si]=z then changes(Z); adds(Z)
                end else if stem[si]=x ∨ stem[si]=s then changes(S) else
                begin if labial(stem[si]) then adds(l); adds(y)
                end
        end softening; transfer;
        if soft stem ∧ finalcons=y then substitutive softening:
        begin if form[fi]=y then omit;
                if form[fi]=t then
                begin if form[fi-1]=s then begin omit; change(S); add(S) end else change(C)
                end else if form[fi]=d ∨ form[fi]=z then
                begin omit; if form[fi]=z then change(Z); add(Z)
                end else if form[fi]=s then change(S) else
                begin if labial(form[fi]) then add(l); add(y)
                end
        end substitutive softening;
        if des[1]>36 then add desinence else accent; phonemics; PUNLCR; output;

        simple desinence:= false; di:=2; des[1]:= if open ∧ soft stem then i else o; des[2]:=S;
        if final=A∧stem[si]=j∧stem[si-1]=u ∨ li-si>1 then stress(des[1]); transfer;
        if form[fi]=k ∧ ¬form[fi-1]=t then
        begin if form[fi-1]=s then begin omit; change(S); add(S) end else change(C)
        end else if form[fi]=g then begin omit; if form[fi]=z then change(Z); add(Z) end else
        if ¬soft(form[fi]) then add(y);
        if des[1]>36 then add desinence else accent; phonemics;
        if form[1]=x∧form[2]=a∧form[3]=t∧fi=6 then PUTEXT(k  xo¢i$  xo¢it$) else begin output; change(t); output end sing;
        PLURAL: change(m); output; change(t); add(y); add(i); output;
        des[1]:= if open ∧ soft stem ∧ ¬finalcons=g then a else u; des[2]:=t;
        if final=A∧stem[si]=j∧stem[si-1]=u ∨ li-si>1 then stress(des[1]); transfer;
        if des[1]>36 then add desinence else accent; phonemics; output;

        IMPERATIVE: simple desinence:= true; if final=A∧stem[si]=j∧stem[si-1]=u then stress(stem[si-1]);
        if final=j∧li-si>1 then begin for h:=1 step 1 until li do form[h]:=lexeme[h]; fi:=li end else transfer;
        if soften∧finalcons=p∧accented∧¬removable accent then begin omit; change(y) end else
        if ¬soft(form[fi]) then add(y);
        if ¬(accented∧¬removable accent∧(vowel(form[fi-1])∨vowel(form[fi-2])∧form[fi]=y) ∨ finalcons=ₒ∧¬final=I) then
        begin di:=1; des[1]:=i; accent
        end else if (¬accented∨final=A)∧finalcons=j then stress last vowel; phonemics; PUNLCR; output;
        add(t); add(y); add(i); output; goto BEGIN
    end
```

164, 165. *brízga-* 'sprinkle', *brížžu*; *gloda-* 'gnaw', *glažú*; *máza-* 'smear', *mážu*;

166. *paxa-* 'plow', *pašú*; *p'isa-* 'write', *p'išú*;

167. *sípa-* 'scatter', *sípl'u*; *kolo-* 'stab', *kal'ú*.

169. The stem is transferred to the array *form*, where the changes that are peculiar to the 1 sg. are carried out.

170. If the final consonant of the stem is soft, it most often changes into a palatal consonant or a cluster (J 2.41). This softening is 'substitutive' in the sense that one soft consonant is replaced by another. Examples:

172, 173. *m'ét'i-* 'mark', *m'éču*; *mst'í-* 'revenge oneself', *mššú*;

174, 175. *s'id'é-* 'sit', *s'ižú*; *jézd'i-* 'drive', *jéžžu*; *groz'í-* 'menace', *gražú*;

176. *v'is'é-* 'hang', *v'išú*;

177. *šum'é-* 'make a noise', *šuml'ú*.

180. If the desinence has acquired the stress in line 132 or 137 already, the procedure *accent* need not be invoked. A new line and carriage return symbol precedes the 1 sg. in the output tape.

181. The 2 sg. desinence is *iš* or *oš* (J 2.121).

182. Verbs with an input stem ending in *ová* or *aváj* stress the desinence under the same conditions as in the 1 sg., see line 132 and 137 respectively. The accent of verbs like *darová-* 'grant', *darúju* is not affected because the suffix *uj* of such verbs has received the stress in line 133 and the condition of the present if clause contains an unstressed *u*. The stem is transferred to the array *form*, where stem-final velars are replaced by palatals and other stem-final consonants undergo 'bare' softening (J 2.42.B):

183, 184. *p'ok-* 'bake', *p'ikú*, *p'ičóš*;

185. *lga-* 'lie', *lgú*, *lžóš*; *žg-* 'burn', *žgú*, *žžóš*;

186. *žda-* 'wait', *ždú*, *žd'óš*; *pas-* 'tend', *pasú*, *pas'óš*; *rva-* 'tear', *rvú*, *rv'óš*.

187. Cf. line 180.

188. The forms *xóčiš*, *xóčit* instead of **xat'íš*, **xat'ít* (input stem *xot'é-* 'want') are irregular (J 3.1). The 3 sg. form is identical with the 2 sg. but for its final phoneme, which is *t* instead of *š* (J 2.121).

189. The 1 pl. and 2 pl. forms end in *m* and *t'i* respectively (J 2.121); they are for the rest identical with the 2 sg. and 3 sg.

190. The 3 pl. desinence is *at* if the 2 sg. ends in *iš* and *ut* if the 2 sg. ends in *oš* [cf. J 2.121] but for the verb *b'ežá-* 'run', which has *b'igút*. (The combination *soft stem* and *finalcons* = *g* does not occur in any other verb, cf. line 140.)

191. Cf. line 182.

192. Cf. 180 and 187.

193. The imperative of verbs with an input stem in *ová* ends in *új*.

194. The imperative of verbs with an input stem in *avái* is identical with the input stem itself, which has been preserved in the array *lexeme*. Other verbs derive the imperative from the present stem.

195. The verbs *sípa-* 'scatter' and *krápa-* 'trickle' lose the epenthetic *l* in the imperative (J 3.1): *síp'*, *kráp'*.

196. A hard consonant undergoes 'bare' softening before the imperative desinence, e.g., *trónu-* 'touch', *trón'*; *p'ok-* 'bake', *p'ik'í*.

197, 198. The imperative desinence is *i* if the verb has a mobile accent (unstressed input stem) or a removable accent, and also if the present stem ends in two consonants; otherwise it is zero. There is one limitation: if the final consonant of the input stem is *j*, the imperative desinence is *i* only if the input stem ends in *jí* (J 2.122), e.g., *tají-* 'conceal', *tají*, but *stojá-* 'stand', *stój*, and *p'j-* 'drink', *p'éj*.

199. If the final consonant of the input stem is *j* and the imperative desinence is zero, the verb form still has to receive an accent. The output is preceded by a new line and carriage return symbol.

200. The plural form is obtained by adding the suffix *t'i* to the generated verb form. When this form has been punched into the output tape, the next input stem is to be read.

3. THE OUTPUT

The program was executed with a list of 150 verb stems on the X8 of the Mathematical Centre, Amsterdam. The results are shown below.

Amsterdam

```
> b'eła-
  b'iłat'      b'iłal   b'iłala   b'iłala   b'iłal'i
  b'igu   b'iłi$   b'iłit   b'iłim   b'iłit'i   b'igut
  b'igi   b'ig'it'i

> b'el'ej-
  b'il'et'      b'il'el   b'il'ela   b'il'ela   b'il'el'i
  b'il'eju   b'il'eji$   b'il'ejit   b'il'ejim   b'il'ejit'i   b'il'ejut
  b'il'ej   b'il'ejt'i

> b'el'i-
  b'il'it'      b'il'il   b'il'ila   b'il'ila   b'il'il'i
  b'il'u   b'el'i$   b'el'it   b'el'im   b'el'it'i   b'el'at
  b'il'i   b'il'it'i
```

> b'er'oʒ–
b'ir'eč b'ir'ok b'ir'igla b'ir'iglo b'ir'igl'i
b'ir'īgu b'ir'iʒo$ b'ir'iʒot b'ir'iʒom b'ir'iʒot'i b'ir'igut
b'ir'igᵀi b'ir'ig'it'i

> b'j–
b'it' b'il b'ila b'ila b'il'i
bju bjo$ bjot bjom bjot'i bjut
b'ej bᵀejt'i

> boja"–
bajatca bajalsa bajalas' bajalas' bajal'is'
bajus' baji$sa bajitca bajimsa bajit'is' bajatca
bojsa bojtᵀis'

> bra–
brat' bral brala brala bral'i
b'īru b'ir'o$ b'irᵀot b'ir'om b'ir'ot'i b'irut
b'irᵀi b'irᵀit'i

> br'ij–
br'it' br'il br'ila br'ila br'il'i
br'eju br'eji$ br'ejit br'ejim brᵀejit'i br'ejut
br'ej br'ejt'i

> brizga–
brīzgat' brizgal brizgala brizgala brizgal'i
brīʒʒu briʒʒi$ briʒʒit briʒʒim briʒʒit'i briʒʒut
brīʒʒi brīʒʒit'i

> bi–
bit' bil bila bila bil'i
budu bud'i$ budᵀit bud'im bud'it'i budut
but' but't'i

> var'i–
var'it' var'il var'ila var'ila var'il'i
var'u var'i$ var'it var'im var'it'i var'at
var'i var'it'i

> v'el'e–
v'il'et' v'il'el v'il'ela v'il'ela v'il'el'i
v'il'u v'il'i$ v'il'it v'il'im vᵀil'it'i v'il'at
v'il'i v'il'it'i

> v'er'i–
v'er'it' v'er'il v'er'ila v'er'ila v'er'il'i
v'er'u v'er'i$ v'er'it v'er'im v'er'it'i v'er'at
v'er' v'er't'i

> v'od–
v'ist'i v'ol v'ila v'ilo v'il'i
v'idu v'id'o$ v'id'ot v'idᵀom v'id'ot'i v'idut
v'idᵀi v'idᵀit'i

> v'oz–
v'ist'i v'os v'izla v'izlo v'izl'i
v'izu v'iz'o$ v'iz'ot v'iz'om v'iz'ot'i v'izut
v'izᵀi v'izᵀit'i

> v'is'e–
v'is'et' v'is'el v'is'ela v'is'ela v'is'el'i
v'isu v'is'i$ vᵀis'it vᵀis'im v'is'it'i vᵀis'at
v'isᵀi v'isᵀit'i

> gas(nu)–
gasnut' gas gasla gasla gasl'i
gasnu gasn'i$ gasn'it gasn'im gasn'it'i gasnut
gasn'i gasn'it'i

> gloda–
gladat' gladal gladala gladala gladal'i
glažu glži$ glžit glžim glžit'i glžut
glažī glažit'i

> gna–
gnat' gnal gnala gnala gnal'i
gan'u gon'i$ gon'it gon'im gon'it'i gon'at
gan'ī gan'it'i

> gn'ij–
gn'it' gn'il gn'ila gn'ila gn'il'i
gn'īju gn'ijo$ gn'ijot gn'ījom gn'ᵀijot'i gn'ijut
gn'ij gn'ijtᵀi

> gnu–
gnut' gnul gnula gnula gnul'i
gnu gn'o$ gn'ot gn'om gn'ot'ī gnut
gnᵀi gnᵀit'i

> govor'i–
gavar'īt' gavar'il gavar'ila gavar'ila gavar'il'i
gavar'u gavar'i$ gavar'it gavar'im gavar'it'i gavar'at
gavar'ī gavar'īt'i

> graf'i–
graf'īt' graf'il graf'ila graf'ila graf'il'i
grafl'u graf'i$ graf'it graf'im graf'it'i graf'at
graf'ī graf'īt'i

> gr'ob–
gr'ist'i gr'op gr'ibla gr'iblo gr'ibl'i
gr'ibu gr'ib'o$ gr'ib'ot gr'ib'om gr'ib'ot'i gr'ibut
gr'ibᵀi gr'ibᵀit'i

> gr'ej–
gr'ēt' gr'el gr'ela gr'ela gr'el'i
gr'ēju gr'eji$ gr'ejit gr'ejim gr'ᵀejit'i gr'ejut
gr'ēj gr'ejt'i

> groz'i–
graz'īt' graz'il graz'ila graz'ila graz'il'i
gražu graz'i$ graz'it graz'im graz'it'i graz'at
grazᵀi grazᵀit'i

> griz–
grīst' gris grizla grizla grizl'i
grīzu griz'o$ griz'ot grīz'om griz'ot'i grizut
grizᵀi grizᵀit'i

> davaj–
davat' daval davala davala daval'i
dajū dajo$ dajot dajom dajot'i dajut
davaj davajt'i

> darova–
daravat' daraval daravala daravala daraval'i
darujū daruji$ darujit darujim darujit'i darujut
daruj darujt'i

> daj–
dat' dal dala dala dal'i
dam da$ dast dad'im dad'it'i dadut
daj dajt'i

> dv'inu–
dv'inut' dv'inul dv'inula dv'inula dv'inul'i
dv'inu dv'in'i$ dv'in'it dv'in'im dv'in'it'i dv'inut
dv'in' dv'in't'i

> d'elaj–
d'elat' d'elal d'elala d'elala d'elal'i
d'elaju d'elaji$ d'elajit d'elajim d'elajit'i d'elajut
d'elaj d'elajt'i

> d'en–
d'et' d'el d'ela d'ela d'el'i
d'enu d'en'i$ d'en'it d'en'im d'en'it'i d'enut
d'en' d'en't'i

> dn'ova–
dn'ivat' dn'ival dn'ivala dn'ivala dn'ival'i
dn'uju dn'uji$ dn'ujit dn'ujim dn'ujit'i dn'ujut
dn'uj dn'ujt'i

> dra–
drat' dral drala drala dral'i
d'iru d'ir'o$ d'ir'ot d'ir'om d'ir'ot'i d'irut
d'ir'i d'ir'it'i

> dr'ema–
dr'imat' dr'imal dr'imala dr'imala dr'imal'i
dr'iml'u dr'eml'i$ dr'eml'it dr'eml'im dr'eml'it'i dr'eml'ut
dr'iml'i dr'iml'it'i

> duj–
dut' dul dula dula dul'i
duju duji$ dujit dujim dujit'i dujut
duj dujt'i

> jożi"–
jożitca jożilsa jożilas' jożilas' jożil'is'
jożus' jożi$sa jożitca jożimsa jożit'is' jożatca
jo$sa jo$t'is'

> jezd'i–
jezd'it' jezd'il jezd'ila jezd'ila jezd'il'i
jeżżu jezd'i$ jezd'it jezd'im jezd'it'i jezd'at
jezd'i jezd'it'i

> jed–
jest' jel jela jela jel'i
jem je$ jest jid'im jid'it'i jid'at
je$ je$t'i

> jexa–
jexat' jexal jexala jexala jexal'i
jedu jed'i$ jed'it jed'im jed'it'i jedut
jet' jet't'i

> żażda–
żażdat' żażdal żażdala żażdala żażdal'i
żażdu żażd'i$ żażd'it żażd'im żażd'it'i żażdut
żażd'i żażd'it'i

> ʑm-
ʑat' ʑal ʑala ʑala ʑal'i
ʑmu ʑm'o$ ʑm'ot ʑmᵀom ʑm'ot'i ʑmut
ʑmᵀi ʑmᵀit'i

> ʑn-
ʑat' ʑal ʑala ʑala ʑal'i
ʑnu ʑn'o$ ʑn'ot ʑnᵀom ʑn'ot'i ʑnut
ʑnᵀi ʑnᵀit'i

> ʑda-
ʑdat' ʑdal ʑdala ʑdala ʑdal'i
ʑdu ʑd'o$ ʑd'ot ʑd'om ʑd'ot'i ʑdut
ʑdᵀi ʑdᵀit'i

> ʑg-
ʑeč ʑok ʑgla ʑglo ʑgl'i
ʑgu ʑʑo$ ʑʑot ʑʑom ʑʑot'i ʑgut
ʑgᵀi ʑg'it'i

> ʑiv-
ʑit' ʑil ʑila ʑila ʑil'i
ʑivu ʑiv'o$ ʑivᵀot ʑiv'om ʑiv'ot'i ʑivut
ʑivᵀi ʑivᵀit'i

> zva-
zvat' zval zvala zvala zval'i
zavu zav'o$ zav'ot zavᵀom zav'ot'i zavut
zavᵀi zavᵀit'i

> znaj-
znat' znal znala znala znal'i
znaju znaji$ znajit znajim znajit'i znajut
znaj znajt'i

> id-
it'i $ol $la $lo $l'i
idu id'o$ id'ot idᵀom id'ot'i idut
idᵀi idᵀit'i

> iska-
iskat' iskal iskala iskala iskal'i
i$$u i$$i$ i$$it i$$im i$$it'i i$$ut
i$$i i$$it'i

> klad-
klast' klal klala klala klal'i
kladu klad'o$ klad'ot klad'om klad'ot'i kladut
kladᵀi kladᵀit'i

> kl'ev'eta-
kl'iv'itat' kl'iv'ital kl'iv'itala kl'iv'itala kl'iv'ital'i
kl'iv'i$u kl'iv'e$$i$ kl'iv'e$$it kl'iv'e$$im kl'iv'e$$itᵀi kl'iv'e
kl'iv'i$i kl'iv'i$it'i

> kl'en-
kl'est' kl'el kl'ila kl'ala kl'al'i
kl'inu kl'in'o$ kl'in'ot klᵀin'om kl'in'ot'i kl'inut
kl'inᵀi kl'inᵀit'i

> kova-
kavat' kaval kavala kavala kaval'i
kuju kujo$ kujot kujom kujot'i kujut
kuj kujtᵀi

> kolo–
kalot' kalol kalola kalola kalol'i
kal'u kol'iś kol'it kol'im kol'it'i kol'ut
kal'i kal'it'i

> krapa–
krapat' krapal krapala krapala krapal'i
krapl'u krapl'iś krapl'it krapl'im krapl'it'i krapl'ut
krap' krap't'i

> krad–
krast' kral krala krala kral'i
kradu krad'oś krad'ot krad'om krad'ot'i kradut
krad'i krad'it'i

> kr'iknu–
kr'iknut' kr'iknul kr'iknula kr'iknula kr'iknul'i
kr'iknu kr'ikn'iś kr'ikn'it kr'ikn'im kr'ikn'it'i kr'iknut
kr'ikn'i kr'ikn'it'i

> kr'iča–
kr'ičat' kr'ičal kr'ičala kr'ičala kr'ičal'i
kr'ču kr'ičiś kr'ičit kr'ičim kr'ičit'i kr'ičat
kr'ći kr'ćit'i

> kroj–
krit' kril krila krila kril'i
kroju krojiś krojit krojim krojit'i krojut
kroj krojt'i

> kup'i–
kup'it' kup'il kup'ila kup'ila kup'il'i
kupl'u kup'iś kup'it kup'im kup'it'i kup'at
kup'i kup'it'i

> lga–
lgat' lgal lgala lgala lgal'i
lgu lžoś lžot lžom lžot'i lgut
lg'i lg'it'i

> l'ožа–
l'ižat' l'ižal l'ižala l'ižala l'ižal'i
l'ižu l'ižiś l'ižit l'ižim l'ižit'i l'ižat
l'ižit l'ižit'i

> l'ez–
l'est' l'es l'ezla l'ezla l'ezl'i
l'ezu l'ez'iś l'ez'it l'ez'im l'ez'it'i l'ezut
l'es' l'es't'i

> l'og–
l'eč l'ok l'igla l'iglo l'igl'i
l'agu l'ažiś l'ažit l'ažim l'ažit'i l'agut
l'ak l'akt'i

> l'j–
l'it' l'il l'ila l'ila l'il'i
lju lješ ljot ljom ljot'i ljut
l'ej l'ejt'i

> l'iši–
l'išit' l'išil l'išila l'išila l'išil'i
l'iśu l'iśiś l'iśit l'iśim l'iśit'i l'iśat
l'iśi l'iśit'i

> l'st'i–
l'st'it' l'st'il l'st'ila l'st'ila l'st'il'i
l'$$u l'st'i$ l'st'it l'st'im l'st'it'i l'st'at
l'st'i l'st'it'i

> l'ub'i–
l'ub'it' l'ub'il l'ub'ila l'ub'ila l'ub'il'i
l'ubl'u l'ub'i$ l'ub'it l'ub'im l'ub'it'i l'ub'at
l'ub'i l'ub'it'i

> maza–
mazat' mazal mazala mazala mazal'i
mazu mazi$ mazit mazim mazit'i mazut
ma$ ma$t'i

> m'ot–
m'ist'i m'ol m'ila m'ilo m'il'i
m'itu m'it'o$ m'it'ot m'it'om m'it'ot'i m'itut
m'it'i m'it'it'i

> m'et'i–
m'et'it' m'et'il m'et'ila m'et'ila m'et'il'i
m'etu m'et'i$ m'et'it m'et'im m'et'it'i m'et'at
m'et' m'et't'i

> molo–
malot' malol malola malola malol'i
m'il'u m'el'i$ m'el'it m'el'im m'el'it'i m'el'ut
m'il'i m'il'it'i

> mst'i–
mst'it' mst'il mst'ila mst'ila mst'il'i
ms$u mst'i$ mst'it mst'im mst'it'i mst'at
mst'i mst'it'i

> mca"–
mcatca mcalsa mcalas' mcalas' mcal'is'
mcus' mci$sa mcitca mcimsa mcit'is' mcatca
mcis' mcit'is'

> moj–
mit' mil mila mila mil'i
moju moji$ mojit mojim mojit'i mojut
moj mojt'i

> moj"–
mitca milsa milas' milas' mil'is'
mojus' moji$sa mojitca mojimsa mojit'is' mojutca
mojsa mojt'is'

> mica–
micat' mical micala micala mical'i
micu mici$ micit micim micit'i micat
mici mici't'i

> n'os–
n'ist'i n'os n'isla n'islo n'isl'i
n'isu n'is'o$ n'is'ot n'is'om n'is'ot'i n'isut
n'is'i n'is'it'i

> nos'i–
nas'it' nas'il nas'ila nas'ila nas'il'i
na$u nos'i$ nos'it nos'im nos'it'i nos'at
nas'i nas'it'i

> ora–
arat' aral arala arala aral'i
ar'u or'i$ or'it or'im or'it'i or'ut
ar'i ar'it'i

> ora–
arat' aral arala arala aral'i
aru ar'o$ ar'ot ar'om ar'ot'i arut
ar'i ar'it'i

> pas–
past'i pas pasla paslo pasl'i
pasu pas'o$ pas'ot pas'om pas'ot'i pasut
pas'i pas'it'i

> paxa–
paxat' paxal paxala paxala paxal'i
pa$u pa$i$ pa$it pa$im pa$it'i pa$ut
pa$i pa$it'i

> p'r–
p'ir'et' p'or p'orla p'orla p'orl'i
pru pr'o$ pr'ot pr'om pr'ot'i prut
pr'i pr'it'i

> poj–
p'et' p'el p'ela p'ela p'el'i
paju pajo$ pajot pajom pajot'i pajut
poj pojt'i

> p'ok–
p'e¢ p'ok p'ikla p'iklo p'ikl'i
p'iku p'i¢o$ p'i¢ot p'i¢om p'i¢ot'i p'ikut
p'ik'i p'ik'it'i

> p'isa–
p'isat' p'isal p'isala p'isala p'isal'i
p'i$u p'i$i$ p'i$it p'i$im p'i$it'i p'i$ut
p'i$i p'i$it'i

> p'j–
p'it' p'il p'ila p'ila p'il'i
pju pjo$ pjot pjom pjot'i pjut
p'ej p'ejt'i

> plaka–
plakat' plakal plakala plakala plakal'i
pla¢u pla¢i$ pla¢it pla¢im pla¢it'i pla¢ut
pla¢ pla¢t'i

> pl'ova–
pl'ivat' pl'ival pl'ivala pl'ivala pl'ival'i
pl'uju pl'ujo$ pl'ujot pl'ujom pl'ujot'i pl'ujut
pl'uj pl'ujt'i

> pliv–
plit' plil plila plila plil'i
pliv'u pliv'o$ pliv'ot pliv'om pliv'ot'i plivut
pliv'i pliv'it'i

> poji–
pajit' pajil pajila pajila pajil'i
paju paji$ pajit pajim pajit'i pajat
paji pajit'i

> poro–
parot' parol parola parola parol'i
par'u por'i$ por'it por'im por'it'i por'ut
par'i par'it'i

> pr'ad–
pr'ast' pr'al pr'ila pr'ala pr'al'i
pr'idu pr'id'o$ pr'id'ot pr'id'om pr'id'ot'i pr'idut
pr'id'i pr'id'it'i

> pr'ata–
pr'atat' pr'atal pr'atala pr'atala pr'atal'i
pr'acu pr'aci$ pr'acit pr'acim pr'acit'i pr'acut
pr'ac pr'act'i

> rost–
rast'i ros rasla raslo rasl'i
rastu rast'o$ rast'ot rast'om rast'ot'i rastut
rast'i rast'it'i

> rva–
rvat' rval rvala rvala rval'i
rvu rv'o$ rv'ot rv'om rv'ot'i rvut
rv'i rv'it'i

> r'ov'e–
r'iv'et' r'iv'el r'iv'ela r'iv'ela r'iv'el'i
r'ivu r'iv'o$ r'iv'ot r'iv'om r'iv'ot'i r'ivut
r'iv'i r'iv'it'i

> r'eza–
r'ezat' r'ezal r'ezala r'ezala r'ezal'i
r'ezu r'ezi$ r'ezit r'ezim r'ezit'i r'ezut
r'es r'est'i

> rza–
rzat' rzal rzala rzala rzal'i
rzu rzo$ rzot rzom rzot'i rzut
rzi rzit'i

> rod'i–
rad'it' rad'il rad'ila rad'ila rad'il'i
razu rad'i$ rad'it rad'im rad'it'i rad'at
rad'i rad'it'i

> ropta–
raptat' raptal raptala raptala raptal'i
rap$$u rop$$i$ rop$$it rop$$im rop$$it'i rop$$ut
rap$$i rap$$it'i

> rub'i–
rub'it' rub'il rub'ila rub'ila rub'il'i
rubl'u rub'i$ rub'it rub'im rub'it'i rub'at
rub'i rub'it'i

> rugaj–
rugat' rugal rugala rugala rugal'i
rugaju rugaji$ rugajit rugajim rugajit'i rugajut
rugaj rugajt'i

> saxar'i–
saxar'it' saxar'il saxar'ila saxar'ila saxar'il'i
saxar'u saxar'i$ saxar'it saxar'im saxar'it'i saxar'at
saxar' saxar't'i

> s'ed–
s'est' s'el s'ela s'ela s'el'i
s'adu s'ad'i$ s'ad'it s'ad'im s'ad'it'i s'adut
s'at' s'at't'i

> s'etova–
s'etavat' s'etaval s'etavala s'etavala s'etaval'i
s'etuju s'etuji$ s'etujit s'etujim s'etujit'i s'etujut
s'etuj s'etujt'i

> s'id'e–
s'id'et' s'id'el s'id'ela s'id'ela s'id'el'i
s'iżu s'id'i$ s'id'it s'id'im s'id'it'i s'id'at
s'id'i s'id'it'i

> skaka–
skakat' skakal skakala skakala skakal'i
skaću skaći$ skaćit skaćim skaćit'i skaćut
skaći skaćit'i

> skr'eżeta–
skr'iżitat' skr'iżital skr'iżitala skr'iżitala skr'iżital'i
skr'iżi$u skr'iże$$i$ skr'iże$$it skr'iże$$im skr'iże$$it'i skr'iże$$ut
skr'iżi$$i skr'iżi$$it'i

> skr'ob–
skr'ist'i skr'op skr'ibla skr'iblo skr'ibl'i
skr'ibu skr'ib'o$ skr'ib'ot skr'ib'om skr'ib'ot'i skr'ibut
skr'ib'i skr'ib'it'i

> sla–
slat' slal slala slala slal'i
$l'u $l'o$ $l'ot $l'om $l'ot'i $l'ut
$l'i $l'it'i

> sm'ej–
sm'et' sm'el sm'ela sm'ela sm'el'i
sm'eju sm'eji$ sm'ejit sm'ejim sm'ejit'i sm'ejut
sm'ej sm'ejt'i

> sm'eja"–
sm'ijatca sm'ijalsa sm'ijalas' sm'ijalas' sm'ijal'is'
sm'ijus' sm'ijo$sa sm'ijotca sm'ijomsa sm'ijot'is' sm'ijutca
sm'ejsa sm'ejt'is'

> sosa–
sasat' sasal sasala sasala sasal'i
sasu sas'o$ sas'ot sas'om sas'ot'i sasut
sas'i sas'it'i

> spa–
spat' spal spala spala spal'i
spl'u sp'i$ sp'it sp'im sp'it'i sp'at
sp'i sp'it'i

> +stavaj–
+stavat' +staval +stavala +stavala +staval'i
+staju +stajo$ +stajot +stajom +stajot'i +stajut
+stavaj +stavajt'i

> stav'i–
stav'it' stav'il stav'ila stav'ila stav'il'i
stavl'u stav'i$ stav'it stav'im stav'it'i stav'at
staf' staf't'i

> stan–
stāt' stal stala stala stal'i
stānu stan'ī$ stān'it stan'im stan'it'i stanut
stān' stān't'i

> stla–
stlāt' stlal stlala stlala stlal'i
st'īl'u st'elᵀī$ stᵀel'it st'el'im st'el'it'i st'el'ut
st'il'ī st'īl'it'i

> stona–
stanat' stanal stanala stanala stanal'i
stanū ston'i$ ston'it ston'im ston'it'ī stonut
stanᵀī stan'it'i

> stoja–
stajāt' stajal stajala stajale stajal'i
stajū staji$ stajit stajim stajit'i stajet
stcj stojtᵀi

> str'ig–
str'īć str'ik str'igla str'igla str'igl'i
str'īgu str'iźo$ str'īźot str'īźom str'īźot'i str'igut
str'igᵀī str'ig'it'i

> stuća–
stućat' stućal stućala stućala stućal'i
stućū stući$ stućit stućim stućit'i stućat
stućī stućit'i

> sipa–
sīpat' sipal sipala sipala sipal'i
sīpl'u siplᵀī$ sīpl'it sīpl'im sīpl'it'i sipl'ut
sīp' sipᵀt'i

> taji–
tajīt' tajil tajila tajila tajil'i
taju taji$ tajit tajim tajit'i tajat
tajī tajīt'i

> t'r–
t'ir'et' t'or t'orla t'orla t'orl'i
tru tr'o$ trᵀot tr'om trᵀot'i trut
trᵀī trᵀit'i

> t'erp'e–
t'irp'et' t'irp'el t'irp'ela t'irp'ela t'irp'el'i
t'irplᵀu t'erp'i$ t'erp'it t'erp'im t'erp'it'i t'erp'at
t'irp'ī t'irp'it'i

> tka–
tkat' tkal tkala tkala tkal'i
tkū tk'o$ tk'ot tk'om tk'ot'ī tkut
tkᵀī tkᵀit'i

> tolknu–
talknūt' talknul talknula talknula talknul'i
talknū talkn'o$ talkn'ot talkn'om talkn'ot'ī talknut
talknᵀī talknᵀit'i

> tolok–
taloć talok talkla talklo talkl'i
talkū talćo$ talćot talćom talćot'i talkut
talkᵀī talk'it'i

> tonu–
tanut' tanul tanula tanula tanul'i
tanu ton'i$ ton'it ton'im ton'it'I tonut
tanTi Tan'it'i

> torżestvova–
tarżistvavat' tarżistvaval tarżistvavala tarżistvavala tarżistvaval'i
tarżistvuju tarżistvuji$ Tarżistvujit Tarżistvujim tarżistvujit'i Tarżistvujut
tarżistvuj tarżistvujt'i

> tr'ep'eta–
tr'ip'itat' tr'ip'ital tr'ip'itala tr'ip'itala tr'ip'ital'i
tr'ip'i$$u tr'ip'e$$i$ tr'ip'e$$it tr'ip'e$$im tr'ip'e$$itTi tr'ip'e$$ut
tr'ip'i$$i tr'ip'I$$it'i

> tronu–
tronut' tronul tronula tronula tronul'i
tronu tron'i$ tron'it tron'im tron'it'i tronut
tron' tron't'i

> tr'as–
tr'ist'i tr'as tr'isla tr'islo tr'isl'i
tr'isu tr'is'o$ tr'is'ot tr'is'om tr'is'ot'i tr'isut
tr'isTi tr'isTit'i

> tr'as"–
tr'ist'is' tr'assa tr'islas' tr'islos' tr'isl'is'
tr'isusT tr'is'o$sa tr'is'otca tr'is'omsa tr'is'ot'is' tr'isutca
tr'isTis' tr'isTit'is'

> uci–
ucit' ucil ucila ucila ucil'i
ucu uci$ ucit ucim ucit'i ucat
uci ucit'i

> uci"–
ucitca ucilsa ucilas' ucilas' ucil'is'
ucus' uci$sa ucitca ucimsa ucit'is' ucatca
ucis' ucit'is'

> xod'i–
xad'it' xad'il xad'ila xad'ila xad'il'i
xażu xod'i$ xod'it xod'im xod'it'i xod'at
xadTi xad'it'i

> xot'e–
xat'et' xat'el xat'ela xat'ela xat'el'i
xacu xoci$ xocit xatTim xat'It'i xatTat
xatTi xat'it'i

> xoxota–
xaxatat' xaxatal xaxatala xaxatala xaxatal'i
xaxacu xaxoci$ xaxocit xaxocim xaxocit'i xaxocut
xaxaci xaxacit'i

> xran'i–
xran'it' xran'il xran'ila xran'ila xran'il'i
xran'u xran'i$ xran'it xran'im xran'it'i xran'at
xran'I xran'it'i

> carapnu–
carapnut' carapnul carapnula carapnula carapnul'i
carapnu carapn'i$ carapn'It carapn'Im carapn'It'i carapnut
carapn'i carapn'it'i

> +ȼez(nu)–
+ȼeznut' +ȼes +ȼezla +ȼezla +ȼezl'i
+ȼeznu +ȼezn'iʃ +ȼezn'it +ȼezn'im +ȼezn'it'i +ȼeznut
+ȼezn'i +ȼezn'it'i

> ȼitaj–
ȼitat' ȼital ȼitala ȼitala ȼital'i
ȼitaju ȼitajiʃ ȼitajit ȼitajim ȼitajit'i ȼitajut
ȼitaj ȼitajt'i

> ʃj–
ʃit' ʃil ʃila ʃila ʃil'i
ʃju ʃjoʃ ʃjot ʃjom ʃjotʸi ʃjut
ʃej ʃejt'i

> ʃum'e–
ʃum'et' ʃum'el ʃum'ela ʃum'ela ʃum'el'i
ʃumlʸu ʃum'iʃ ʃum'it ʃum'im ʃum'it'i ʃum'at
ʃum'i ʃum'it'i

>