

The following assumes that the reader is familiar with the broader outline of my proposed project as set out in my Oxford application-documents, specifically the main research proposal and/or either of the two UKRI funding application forms submitted alongside it.

The Autoreconstructor as a program is largely unchanged since it was first done in 2021, when I was very new to programming, so as code it is pretty atrocious, but I've done a lot more foreign-language-focused programming since then and would easily be able to redo it more sensibly. The full code is viewable at <https://github.com/12401453/Autoreconstructor>.

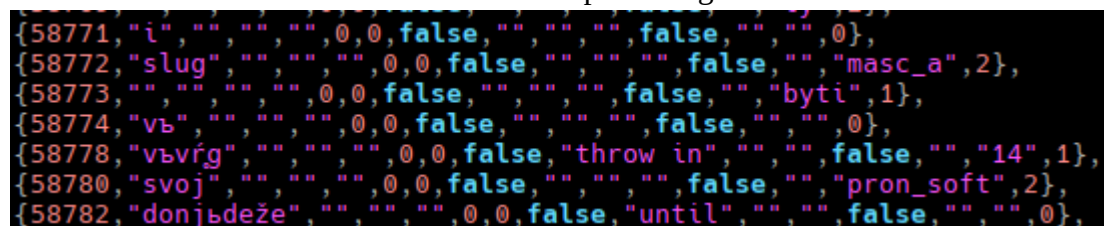
### Why do I need an Autoreconstructor in the first place?

My goal is to enable quick, easy and comprehensive analysis of manuscript spellings, so I need to produce LCS reconstructions of every form in the texts such that they will show up in searches for specific LCS phonemes or phoneme-class sequences (a crude search-mechanism for the already-reconstructed texts can still be found at <http://slavtexts.infinityfreeapp.com>). Doing this by hand isn't really that much effort, but since the TOROT corpus contains high-quality lemmatisation and 10-place morphology-tagging, it's more fun to take the list of TOROT lemmas, reconstruct for each the invariant part of the stem[s], label each one for inflectional-category, then just stick the correct inflections onto the end of the correct stems according to what the morphology tag tells us.

### How does it currently work?

The reconstructed LCS lemmas (and, if necessary, stems) and their inflection-classes, as well as a lot of other possibly relevant information (loanword-origin and source-language, whether something is a blatant non-integrated post-LCS loan, whether a differently-formed doublet of the stem occurs which TOROT has no separate lemmatisation for, etc.) are held in a sort of 'master-spreadsheet'

([https://docs.google.com/spreadsheets/d/1qw75jk\\_oLbjohmdh\\_qlC6B1yb9xRhqwqY](https://docs.google.com/spreadsheets/d/1qw75jk_oLbjohmdh_qlC6B1yb9xRhqwqY)), which started life just as a list of all the TOROT Church-Slavonic lemmas. To fully reconstruct the Marianus and (TOROT's part of) Euchologium Sinaiticum, I just filtered the lemmas to only display those which occur in Mar. and Euch., then manually reconstructed the stems and added the various pieces of auxilliary information. The necessary parts of the spreadsheet then get converted into a C++ std::set data-structure of what I've called Lemma structs, and it's from this std::set that the program retrieves the correct Lemma struct and inspects its stem-form and inflection-class fields to start producing an LCS form.<sup>1</sup>

A screenshot of C++ code defining Lemma structs. The code is color-coded with a dark background. It shows several rows of structs, each representing a lemma. The third row, corresponding to the lemma 'byti', shows a blank stem and the inflection class 'byti'.

```
{58771, "i", "", "", "", 0, 0, false, "", "", "", false, "", "", 0},  
{58772, "slug", "", "", "", 0, 0, false, "", "", "", false, "", "masc_a", 2},  
{58773, "", "", "", "", 0, 0, false, "", "", "", false, "", "byti", 1},  
{58774, "vъ", "", "", "", 0, 0, false, "", "", "", false, "", "", 0},  
{58778, "vъvrg", "", "", "", 0, 0, false, "throw in", "", "", false, "", "14", 1},  
{58780, "svo]", "", "", "", 0, 0, false, "", "", "", false, "", "pron_soft", 2},  
{58782, "donjъdeže", "", "", "", 0, 0, false, "until", "", "", false, "", "", 0},
```

If you look at the third row you can see that the stem of *byti* is blank and its inflection-class is 'byti', because it can't easily be formed from "stem+endings" and needs its own whole

---

<sup>1</sup> Compiling all this data directly into the C++ program (or using C++ at all for this task) rather than reading it off of disk at runtime is an extremely stupid way of doing things and can cause problems with compilation.

paradigm. Indeed, the number of separate inflection-classes I use is pretty obscene for something which claims to be “automatic”: I counted 47 verb-classes and 40 nominal-classes (and I haven’t even added \*d̥ti yet because it doesn’t occur in Mar. or Euch.), but there are actually more class-names than inflection-tables because, for example, all \*o- and \*a-stem nouns and all hard-stemmed adjectives get passed to the same list of 63 (3 numbers \* 7 cases \* 3 genders) endings. Similarly, I reuse the pronoun \*jъ’s table for the long-adjectives and participles.

The endings themselves and broad categorisation into classes are based mostly on Diels (1963) and Olander (2015), who were respectively my ЖЪЗЛЪ and my ПАЛИЦА, but since my goal was to produce exact LCS cognates of the actual text-forms, I had to use much more fine-grained categories than they do. For example, Diels’ verb-class 1.1 includes those with essentially ablauting syllabic-liquids as the vowel of their stop-ending stems, e.g. \*obvelhi and \*vъverhi, which have differing distributions of full- and reduced-grade ablaut, e.g. Mar. 3sg. aorist ОБЛѢУѢ < \*obvelče vs ВЪВРЪЖЕ < \*vъvřže, so I mark them as separate categories to make it easier to apply these ablauting-rules.

The way I read the morphology-tag is very boring and not really relevant, but if you want to see it you can just look at the massive switch-statement at the top of this file [https://github.com/12401453/Autoreconstructor/blob/main/changeLemma\\_field.h](https://github.com/12401453/Autoreconstructor/blob/main/changeLemma_field.h). From a numerified-version of the morphology-tag I compute a number that corresponds to the row of the inflection-table that the given form should select its endings from, e.g. for a verb it will be between 1 and 44, while for a nominal-class it’d be between 1 and 63 (though gender-specific nominal classes would only use a third of the 1-63 range, e.g. the table for the feminine R-stem nouns only has entries from 22-42). Which inflection-table to look into is determined by the inflection-class label which is contained in the massive std::set of Lemma structs as described above.

### What about morphological innovations? Case study: class 1 verbs

My inflection-classes 11, 14 and 15 correspond to subsets of Diels (1963:244-251)’s class 1.1 verbs, i.e. these are class 1 verbs which add their endings directly onto a consonantal-stem with no intervening jot. They differ from my classes 12 and 13, which are respectively nasal and liquid-stemmed, in that their 2nd and 3rd sg. bare aorists add \*-e rather than nothing: \*nese vs \*priĵn > \*priĵe > (in OCS also with a -тъ 3rd sg. marker added) приѡ[тъ], \*umer > оумрѣ[тъ]).

I store the inflection-tables in a C++ container called std::unordered\_map, which is basically a list of key-value pairs<sup>2</sup>, and for each inflection-class’s key I use integers that end in the

---

2 Look-ups in a map have what they call constant time-complexity (O(1)), meaning it’s just as fast no matter how many key-value pairs the map contains, because rather than test each key for equality with the looked-for key until you find it, they use a hashing-function which converts the key to a pointer (memory address) directly to that key’s value-data, meaning the lookup-time is always just equal to the time taken to compute the hash. The same thing happens when you have an Index on a database-table column, which is why looking things up by a table’s PRIMARY KEY column is fast even with millions of rows in your table, whereas lookups by non-indexed rows get progressively slower as the table grows.

number '1' and are at least 10 away from the next class's key. The inflection-table pointed to by the key ending in '1' stores the 'correct' LCS endings, and then deviant endings are stored in supplementary tables pointed to by subsequent integers which end on '2', '3' etc.

```
home > joe > Desktop > Autoreconstructor > outer_verb_map.cpp
1  std::unordered_map<int, inner_map> verb_ = {
2
3      {111, v_11_c0},
4      {112, v_11_c1},
5      {113, v_11_c2},
```

For example, above we can see the tables for the class 1 verb endings (my classes 11, 14 and 15 only): the table indexed by the key 111 contains the 'bare' aorists such as those found in Marianus, like 3rd pl. *пaдѣ*, 3rd sg. *нече*, 1sg. *придѣ*, while the second table (key 112) contains the secondary S-aorists in \*-ox- (Zogr. *пaдoшa* etc.), then the third table (key 113) contains primary S-aorist endings, to produce things like Mar. 1sg. *рѣхъ*, 3sg. *вѣзнѣca*:

```
inner_map v_11_c1 = {
{6, "eta"},
{10, "oxъ"},
{13, "oxově"},
{14, "osta"},
{15, "oste"},
{16, "oxomъ"},
{17, "oste"},
{18, "ošę"},
{24, "ěašeta"},
{37, "qtj"},
};

inner_map v_11_c2 = {
{10, "sb"},
{13, "sově"},
{14, "sta"},
{15, "ste"},
{16, "somъ"},
{17, "ste"},
{18, "se"},
};
```

The real dirt and grime of the Autoreconstructor's current extremely amateurish implementation is in how these different morphological alternatives are detected: I essentially just have banks and banks of if-statements that inspect the last few letters of a cleaned-up version of the TOROT form, if the inflection-class and inflection-table row-number (as computed from the morphology-tag) correspond to desinences which often get replaced in my target texts (hence currently it only tries to detect OCS-specific deviances).

```
if ((conj_type == "21" || conj_type == "11" || conj_type == "jъti" || conj_type == "14" || conj_type ==
"15" || conj_type == "rěsti") && (row_no == 10 || row_no == 13 || row_no == 16)) {
    if (Sniff(cyr_id, "ox", 5)) outer_map_no++;
}
if ((conj_type == "21" || conj_type == "11" || conj_type == "jъti" || conj_type == "14" || conj_type ==
"15" || conj_type == "rěsti") && (row_no == 14 || row_no == 15 || row_no == 17)) {
    if (Sniff(cyr_id, "oct", 4)) outer_map_no++;
}
if ((conj_type == "21" || conj_type == "11" || conj_type == "jъti" || conj_type == "14" || conj_type ==
"15" || conj_type == "rěsti") && row_no == 18) {
    if (Sniff(cyr_id, "ow", 3)) outer_map_no++;
}
```

Here you can see the detection of (among others) these innovated class 1 aorist forms: all the Sniff() function does is chop off the specified number of letters from the end of the Cyrillic form and check to see if it contains the letters passed to it as its second argument<sup>3</sup>. If it does, then Sniff() returns 'true' and the integer-key of the current inflection-table is incremented by one (111 + 1 = 112), causing it to instead select the deviant endings stored in the supplementary table.

<sup>3</sup> It actually just chops off double the specified number of bytes (with each Cyrillic character (usually) being 2-bytes long in UTF8), because I in my infinite wisdom chose to use a language with no inbuilt Unicode support to make a program that does nothing but non-ASCII string-manipulation.

Once this post-processing is done I can write the LCS forms out to a file; often I add it to the original TOROT input .csv file as an additional column, and this gives us a completely regular and predictable index of every word in the TOROT text for which the LCS form has been correctly reconstructed, which makes searching for specific forms extremely easy (as long as you are familiar with my reconstruction-decisions and conventions, as outlined here <http://slavtexts.infinityfreeapp.com/principles.php>).

## Annotation-mistake detection

66370	избавынаѣшта	ἰζβᾰvēḥḂ
67235	възвращаѣтъ	vъzvōrḥḂjetъ
58936	идѣшта	ἰdōḥḂ
60020	поѣшта	pojēmōḥḂ

The second instance is a more extreme mistake, since we have ‘sing’ lemmatised as ‘grab’, as is clear from the context<sup>5</sup>:

5 Note also TOROT's baffling sporadic replacement of Supr.  $\Delta$  by the hooked Glagolitic nasal  $\epsilon$ , which actually only occurs in Zogr. and Mar. (ѡѡѡѡѡѡ, ꙗѡѡѡ etc.) and which Kortlandt (1979?) says is a remnant of nasalised \*y corresponding to the NSL. \*-a endings on masc. Nsg. hard-stemmed class I PRAPs. Olander (2015:89-90) has more discussion, though beware his oversimplifying characterisation of it as “ $\epsilon$  with a special sign” (Велчева and Trubetzkoy are crying out in pain).

```
и́,и,1153752,58771,80991,C-,-----n
бѣаста,быти,1153753,58773,80991,V-,3diia----i
въ,въ,1153754,58774,80991,R-,-----n
огни́,огнь,1153755,60190,80991,Nb,-s---ml--i
по́жста,по́жти,1153756,60020,80991,V-,dppamn-si
и,и,1153757,58771,80991,C-,-----n
хва́лѣшта,хвалити,1153758,67696,80991,V-,dppamn-si
бога,богъ,1153759,60009,80991,Nb,-s---mg--i
```

If we converted these erroneous reconstructions to the regular but unattested ‘normalised’ OCS system which is sometimes employed, i.e. *избавациа, поимѣциа* (a relatively trivial task, see below for an example of me doing it), then the “edit-distances” between the normalised-OCS forms and a cleaned-up version of the Suprasliensis forms should easily be large enough to pass whatever threshold we might set for alerting us of possible TOROT mistakes.

## Other advantages to a comprehensive LCS reconstruction

There are of course a number of problems and difficulties in producing accurate LCS forms which I haven't yet overcome, and a reasonably complete list of them can be found in the "stuff that doesnt work.txt" file which I have appended to the end of this document. I'd like here to focus briefly on one such difficulty, namely problems caused by variations in the forms of verb-prefixes, because it gives a flavour of the benefits that come from being forced to think comprehensively about the recent pre-history of a whole text, rather than just a few specific "interesting" forms:

Some OCS prefixed verbs can vary in whether they have been re-formed, with their original prefix replaced by the analogically-*-jer*-extended preposition, e.g. ὀχλῶδιτι <\*obxoditi / обзлѡдити < \*объ+\*xoditi, Mat. 7 Mar. ወይዘራንታልና, ወይዘሩ < \*obese vs Zogr.

More strikingly, all vowel-stem verbs prefixed with *npě-* must have been either formed or reformed since the TORT metathesis, because an LCS word like *\*perɛľ* could never have given rise to a form like Supr. *прѣлѣ*, which is pretty surprising when you think about how recently the metathesis should've happened relative to our earliest texts. This tells us something either about the timing of the derivation of these prefixed verbs (the *\*per-* prefix is used across N. and E. Slav. in a native guise so it can hardly be that the prefixation occurred AFTER metathesis, but if prefixation occurred when it was still *\*per-* then how come we see no traces whatsoever of etymologically regular *\*\*перидѣтъ*, *\*\*перѣмъше?*) or about their possible artificiality (but this is also unlikely given native ESL. *pepe-* verbs). I dodge the question by just reconstructing it as *\*per+j̥ti*, with the plus-sign indicating that the two parts that gave rise to the attested form could not actually have existed together at the time of the reconstruction (the sequence *\*rj* is illegal under my LCS system (all such earlier clusters are given as *\*ř*), and anyway *\*iti* would never have taken a prothetic *\*j-* were it preceded by *\*per-*).

This problem of reconstructing the \*prě- forms is annoying and boring, but it does serve to highlight the value of my “reconstruct every single word of the text” approach, because we’ve learnt (or been reminded of) at least something about the language in the unwritten period before OCS, namely either that some kind of mass-analogy event must have occurred in the \*per- prefixed verbs after metathesis, or (less likely) that \*per- prefixed verbs did not

yet exist as single phonological words at the time of the metathesis, which would surely have consequences also for people interested in even the semantic side of the history of Slavic aspect and verb-prefixation.

Another potentially cool side effect of having such an Autoreconstructor is it could be combined with automatic lemmatisation and morphological-tagging (which Hanne can already do surprisingly effectively and which I think could possibly be improved further by more careful and targeted data-cleaning and potentially also neural-network approaches like the one described here [Morpheus reference]), to produce an LCS reconstruction straight from a raw text, potentially even a raw manuscript if Transkribus can be trained well enough (though that probably is pushing our luck). Imagine a world where a student could just point their phone at a past exam-paper or homework commentary-passage and get given back a bullet-point list of all the relevant phonological features in the text such that they could completely cheat: if I am successful in producing high-quality comprehensive phonological-annotation then all that would be necessary to achieve this is OCR good enough to locate the printed passage in my database, but it would be interesting to see if such cheating could also be enabled using nothing but the raw text.

### Reproducing normalised OCS from the LCS reconstructions

My LCS system is close enough to attested OCS that one can almost completely flawlessly transform my LCS forms back into the non-existent “normalised” form of OCS which is sometimes presented in grammars and textbooks (essentially an  $\bar{A}$  > /a/ dialect with phonemic /j/, but with fully preserved jers. No such dialect is attested because the only ms. with unfallen jers is the Kiev Folia, which indubitably reflects an  $\bar{A}$  > /ě/ dialect with no /j/).

For this I used a node.js script (attached), which takes as its input the LCS reconstructed forms, and the results from a random section of Marianus are shown below:

TOROT form	Autoreconstructed LCS	Autonormalised OCS
реѣ	reče	реѣ
вѣ	vъ	вѣ
истѣинѣ	jъstinъ	истинѣ
голѣ	golgoľ	глаголиѣ
вѣмѣ	vamъ	вѣмѣ
ѣко	ako	ѣко
вѣдовѣца	vъdovika	вѣдовѣца
си	si	си
оубогаѣ	ubogajĀ	оубогаѣ
боѣ	boĭe	боѣ
вѣѣѣѣ	vъѣѣѣѣ	вѣѣѣѣ
вѣѣѣѣѣ	vъvĭže	вѣѣѣѣѣ

The big problems come when you try to do PV3, because this sound-change has just not been consistently carried out: the regex I use to search for the conditioning environments is  $/[ɨi][kɡx][auq]/^6$ , and while, for example,  $\bar{*}ka$  groups pretty consistently get palatalised (in

---

<sup>6</sup> groups ending in  $\bar{*}o$  only occur in nominal desinences ( $\bar{*}sĭdъko$  etc.), so the Autoreconstructor already applies fronted endings to them and we thus re-palatalise their velars via the much simpler and more reliable PV2 routine instead



OCS at least), the majority of \*ika groups are Gsg. masc. o-stem endings of words like оученикъ, which, if they ever did undergo PV3, pretty quickly levelled it away by analogy with the rest of their paradigms:

```
porzdnika|праздница
narikaјемѣјемъ|нарицаѣмѣмъ
narikaјемоје|нарицаѣмоѣ
učenika|оученица
učenika|оученица
prikasaji|прицасаи
narikaјемъъ|нарицаѣмъи
narikaјемъъ|нарицаѣмъи
[joe@joe-asus-x555l scripts]$ ./LCS* mar_lcs.csv | grep "ika"
```

Here the only correctly palatalised forms are those of the verb нарицати.

The advantages, especially pedagogical, of being able to auto-normalise an OCS text are numerous, but even in failure, as above, we are at least once again forced to confront the reality of PV3 being a very dodgy sound-change, and to reconsider its commonly-assumed conditioning-environment. With the complete data at our fingertips we might even be able to come up with a better explanation for it (e.g. we can see explicitly that preceding \*ъ is far better at triggering it than \*i).

### Caveats

In closing I should just mention that the size of the corpus of texts which interest me (canonical OCS + early East Slavic, and maybe some other early stuff like the Freising Fragments) is small enough that automated techniques are only ever envisaged as a starting point for manual correction: I don't even trust Jagić's editions and fully intend to read every single text I study (with the possible exception of Psal. Sin. which is a nightmare) cover-to-cover from manuscript-photographs before I sign off on any reconstructed forms. I'll never be able to predict and detect every single morphological deviance occurring across such a wide dialectal spectrum of texts, and there'll always be TOROT tagging- and lemmatisation-mistakes, so the point of what has been described above is really just as a way to get the bulk of the easy work done so I can concentrate on hand-correcting the tricky bits. Ёже естъ писано въ пророцѣхъ: се азъ посълихъ а́нѣлъ мои прѣдъ лицемъ твоимъ, иже оутѣтитъ пѣть твою.

### Appendix 1: stuff that doesnt work.txt

--the deviant lengthened-grade stem in the class 5.2(abl) PAP stem of \*ръsati (писавъ etc.) isn't detected (all other deviant lengthened-stems of this verb are).

--метати and искати are always conjugated as class 5.1 or 5.2; the class 5.4 reformations are not detected

--it deliberately doesn't try to detect 3rd dual -te levelled to -ta for root-aorists, because the priority here is detecting secondary sigmatic -ост-, so Supr. 3rd dual. обрѣоста outputs: obrētoste|obrētete|

--Doesn't detect отымати/отимати, сънимати etc. for original \*отъмати etc., or объяти etc. for обати (because I need a better strategy for dealing with stem-deviances).

--comparative мънѣи needs to be checked for as it's not derivable from \*тънѣъ

--дивѣше is flagged as deviant because the stem ends in -ив (and because of the possibility of articulated forms of PAPs I have to check at least the last 8 letters for \*-iv-, meaning this form would always get false-flagged)

--doesn't find pronominal мъногѣ inflections (it easily could)

--deviances in stems (e.g. въсѣдръжителѣ-type nouns) are mostly ignored (though here my decision to mark PV3-dependent frontings as deviances is possibly unjustifiable).

--ниже, скорѣе, мъножѣе, and often лише are tagged as comparatives of adverbs so do not get correctly reconstructed

--ѣт-verbs are not checked for the \*-ох- aorists, only ruki-violating S-aorists, so Supr. процвѣтоша gets reconstructed as prokvišę|prokvišę

--Supr. past passive participle закланъ is not detected, but this is an abhorrent deviance and the regular заколенъ occurs also

--aorist \*boršę (браша) is flagged as a deviant form of \*boršę, but /r/ is a ruki consonant

--Supr. Gsg. личеса/личесе

--Deviances in the participle flexions like fem. Nsg. \*hĀ instead of \*hi, or PAP Npl. masc. \*ši instead of \*še, are detected and changed but the etymologically regular forms aren't generated (due to complications in how participles are passed through the conjugation-pipeline).

--Doesn't bother with the že on nikotorъjъ-že, because this is very rare word and arguably wrongly lemmatised anyway (котеръи spellings)

--the об variant of preposition \*o is not detected (could easily be)

--\*goręti is reconstructed as a normal class 3.1 so the class (5.2? 1?) PRAPs \*goręh- (Ru. горячее), masc. Nsg. \*gorę (if we believe the soft-sign mark), possibly class 1 \*goręh- if this is not just dispalatalised /r/, is ignored

--Doesn't pick up господевѣ, only господю

--божий and велий are constructed as \*božj-, \*velj- + the short-form soft adjective ending without regard to the strength marking, because there are too many instances of these words being marked as weak where no article is detectable, e.g. Мар. велиемъ|велии|541370|-s---oipwi. There are examples of articulated-forms of such words, i.e. those mentioned by Diels 1963 p.198, but the TOROT annotations are not a sensible way to detect them, so until I can be bothered to write a way to check for them, all such articulated forms will be incorrectly reconstructed as short-forms.



--Pronominals with deviant adjective endings like Marianus *adherens* are not detected