Historical-Comparative Reconstruction in Finite-State Technology

James Kilbury , Katina Bontcheva, Natalia Mamerow, Younes Samih

Heinrich-Heine-Universität Düsseldorf, Institut for Language and Information,
Department of Computational Lingistics, Universitätsstr. 1
40225 Düsseldorf, Germany
{kilbury, bontcheva, mamerow, samih}@phil.uni-duesseldorf.de

Computational linguistics (CL) provides tools for many aspects of work in historical linguistics. Of course, CL tools offer invaluable help in preparing and searching text editions, in systematizing data, and many other tasks. Since early CL it has been possible to program replacement rules for sound changes and thus to simulate the derivation of later phonological forms from historical antecedents. Discussions by Kaplan & Kay (1994) and Karttunen (1995, 1997) have led to a greatly refined understanding of the semantics of replacement rules and now provide a theoretical basis for recent finite-state technology (FST) which allows a direct encoding of replacement rules. Not only the "downward" derivation of later forms from antecedents can be computed, but also efficient "upward" derivation of possible antecedents for given later forms. This corresponds directly to the model of historical sound change in terms of ordered replacement rules, which continues to receive wide acceptance in historical linguistics.

The work of Kay (1987) on multi-tier analysis of Arabic morphology with *n*-tape transducers offers a framework in which historical-comparative reconstructions (HCR; cf. Hoenigswald 1966) can easily be expressed formally. Phonological representations of forms in genetically related languages can be encoded as strings on the tapes of a transducer, and regular sound correspondences can be viewed as vectors of segments, each position in a vector corresponding to one of the tapes. Moreover, an additional tape can be populated with symbols chosen as phonetically plausible names of regular correspondences. Strings of such symbols then make up the forms of the postulated proto-language from which the forms of the genetically related daughter languages are assumed to derive.

Problems in the use of *n*-tape transducers (NTTs) as a model for HCR soon became apparent. Undesirable formal properties of NTTs (cf. Wiebe 1992) led to doubts about their suitability as a basis for practical software, which indeed was effectively unavailable. Replacement rules for regular sound changes had no obvious interpretation in the framework.

Recently, fresh attention has been given to the simulation of NTTs within current FST, in particular, for the description of Arabic (cf. Hulden 2009b). In this approach the strings of NTTs are represented by corresponding sections of a single string on a single tape in accord with constraints formulated in predicate logic. The interpretation of HCR and replacement rules relating the tapes to each other, however, remains unclear in this framework.

The approach of our own work is different and presents a method for encoding HCR viewed in terms of NTTs within current formalisms such as **`(x)fst`** (cf. Beesley & Karttunen 2003) and **`foma`** (cf. Hulden 2009a) using replacement rules. The forms of a reconstructed proto-language are encoded as the strings of an upper language. Strings representing forms of the daughter languages collectively populate the lower language, but each string of a daughter language contains a tag designating the respective language. Replacement rules conditioned by language tags map the upper, proto-language into the lower language constituting the union of the daughter languages.

For a simple example we take a fictitious proto-language *X*, which we define as follows:

```
define Vowel   [i, e, a, o, u] ;
define Cons    [p, t, k, s, m, n, r] ;
define ProtoLg [X Cons Vowel Cons (Vowel)] ;  # the phonotactics of X
```

Note that each string of *X* begins with the language tag '**`X`**'.

Sound changes recoded in rules **`r1`** and **`r2`** capture the development from *X* to the daughter language *A*, while **`r3`** derives the forms of daughter language *B*:

```
define r1 [ k -> c || _ [e|i] ] ;  # palatalization of k to c
define r2 [ [e|o] -> a ] ;         # merger of mid vowels e, o with a
define r3 [ Vowel -> 0 || _ .#.] ; # deletion of final vowels
```

The reconstructed historical phonology of the family can then be formulated as follows:

```
define HistPhon [ ProtoLg .o.
                  [ [[X -> A] .o. r1 .o. r2] |
                    [[X -> B] .o. r3       ]  ] ] ;
```

If the network defined by **HistPhon** is on top of the **xfst** or **foma** stack, then **apply down** with the string **Xkepi** produces **Acapi** and **Bkep** in the lower language, while **apply up** with **Akap** produces **Xkap** and **Xkop** as possible antecedents in the upper language.

The software of **(x)fst** and **foma** provides no direct way to test a given set of daughter forms to see whether it has one or more possible antecedents in the proto-language. Using the notation *R*.u to designate the upper language of a relation *R*, however, we simply need, in effect, to compute **apply up** for the individual daughter forms and then take the intersection of the results. Thus,

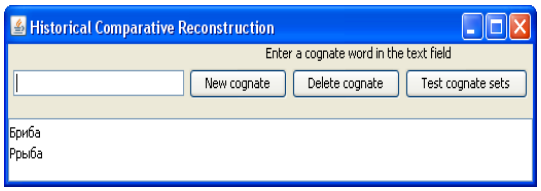
```
read regex [ [HistPhon .o. {Acapi}].u  &  [HistPhon .o. {Bkep}].u ] ;
print words
```

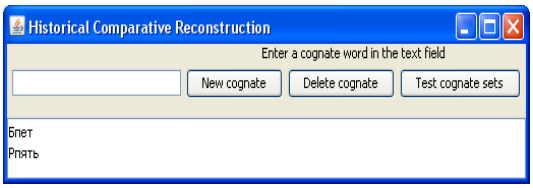
produces the string **Xkepi** as the unique possible antecedent . On the other hand, if, e.g., **Bkipi** is substituted for **Bkep** in this example, then no common antecedent can be computed for the given reconstruction.

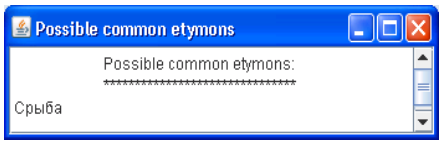
The approach outlined here does not depend on a tree model of genetic relationship, although that has been chosen for the example. Instead, we can capture a wave model or network relationship by introducing rule tags such as **§r1**, **§r2**, etc., together with the notation '**$§r1 _**' for "rule tag **§r1** contained in the left environment of rule **r1**", as follows:

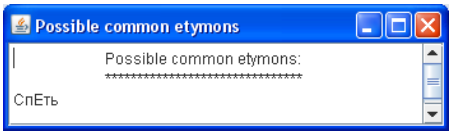
```
define r1       [ k -> c || $§r1 _ [e|i] ] ;
define HistPhon [ ProtoLg .o.
                  [[X -> A §r1 §r2] | [X -> B §r3]] .o.
                   r1 .o. r2 .o. r3                    ] ;
```

The central result of this work is that historical linguists can now use current finite-state based software to test comparative reconstructions of language families and to compute proto-forms from proposed cognate sets. We have tested the technique with data from Slavic and Germanic languages. For Slavic we extracted over 700 cognate sets of Russian and Bulgarian words from Vasmer's etymological dictionary (1953-1958) and compared our reconstructions with the Old Church Slavonic forms provided by Vasmer. Moreover, we have built a Java GUI to facilitate the input and display of Unicode characters. Here is an example for Slavic (where '**Б**', '**Р**', '**С**' are tags for Bulgarian, Russian, and Slavic, and '**Е**' transliterates '**Ѣ**'):









References

Beesley, K. & Karttunen, L. (2003): *Finite-State Morphology*. Stanford: CSLI.
Kaplan, R. & Kay, M. (1994): Regular Models of Phonological Rule Systems, *Computational Linguistics* 20: 331-378.
Karttunen, L. (1995): The replace operator. In: *33rd ACL Proceedings*, 16-23.
Karttunen, L. (1997): The replace operator. In: E. Roche & Y. Schabes, *Finite-State Language Processing*. Cambridge, Mass. & London: MIT Press.
Kay, M. (1987): Nonconcatenative Finite-State Morphology. In: *Proceedings of the EACL 2009*, 2-10.
Hoenigswald, H. M. (1966): *Language Change and Linguistic Reconstruction*. Chicago: UC Press.
Hulden, M. (2009a): Foma: a finite-state compiler and library. In: *Proceedings of the EACL 2009 Demonstrations Session*, 29-32.
Hulden, M. (2009b): *Finite-State Machine Construction Methods and Algorithms for Phonology and Morphology*. Dissertation, University of Arizona.
Vasmer, M. (1953-58): *Russisches etymologisches Wörterbuch*. Winter, Heidelberg .
Wiebe, B. (1992): *Modelling Autosegmental Phonology with Multi-Tape Finite State Transducers*. M.S. thesis, Simon Fraser University.