Historical-Comparative Reconstruction in Finite-State Technology

James Kilbury
Katina Bontcheva
Natalia Mamerow
Younes Samih

Ninth International Tbilisi Symposium on Language, Logic and Computation
Kutaisi, September 26-30, 2011

Outline

- Background: tools and methods
- Motivation for upward tracing
- Cogs & Etyms: architecture of the tool
  - the language-description module
  - the modules for automatic writing of complicated regular expressions
  - the Java Gui
- Testing

Background: tools

Beesley, K. R., Karttunen, L. (2003): *Finite State Morphology*. CSLI, Stanford

Hulden, M. (2009a): Foma: a finite-state compiler and library. In: *Proceedings of the EACL 2009 Demonstrations Session*, pp. 29-32.

Hulden, M. (2009b): *Finite-State Machine Construction Methods and Algorithms for Phonology and Morphology*. Dissertation, University of Arizona.

Background: methods

Hoenigswald, H. M. (1966): *Language Change and Linguistic Reconstruction*. Chicago: UC Press.

Karttunen, L. (1995): The replace operator. In: *33rd ACL Proceedings*, 16-23.

Kaplan, R. & Kay, M. (1994): Regular Models of Phonological Rule Systems, *Computational Linguistics* 20: 331-378.

Kay, M. (1987): Nonconcatenative Finite-State Morphology. In: *Proceedings of the EACL 2009*, 2-10.

Wiebe, B. (1992): *Modelling Autosegmental Phonology with Multi-Tape Finite State Transducers.* M.S. thesis, Simon Fraser University.

Motivation for upward tracing

- 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. 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 (NTTs) offers a framework in which historical-comparative reconstructions (HCR) can easily be expressed formally. However, problems in the use of NTTs as a model for HCR soon became apparent.

Motivation for upward tracing

- 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 xfst and foma using replacement rules.

- Our main goal is to provide historical linguists with a tool that allows them to use current finite-state based software to test comparative reconstructions of language families and to compute proto-forms from proposed cognate sets.

Our approach to upward tracing

- The forms of a reconstructed proto-language are encoded as the strings of an upper language. The strings contain a tag that designates the proto-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.

- Here are some strings of the fictitious proto-language X from our example (cf. slide 15) that populate the upper language:
  - Xpik, Xpek, Xpak, Xpok, Xpuk, Xpaki, Xpake, Xpaka, Xpako, Xpaku

- Here are some strings of the daughter languages of X that collectively populate the lower language:
  - Apik, Apak, Apuk, Bpik, Bpek, Bpak, Bpok, Bpuk, Apaci, Apaca, Apaka

Our approach to upward tracing

- Sound changes are encoded in rules that capture the development from the proto-language to the daughter languages:

```
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
```

Our approach to upward tracing

- Replacement rules conditioned by language tags map the upper, proto-language into the lower language constituting the union of the daughter languages:

```
[[X -> A] .o. r1 .o. r2]
[[X -> B] .o. r3       ] ]
```

Our approach to upward tracing

- If the network defined for the historical phonology description in our example 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.

$$\begin{tikzcd}
 & \text{Xkepi} \arrow[dl] \arrow[dr] & & \text{Xkap} & & \text{Xkop} \\
\text{Acapi} & & \text{Bkep} & & \text{Akap} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Our approach to upward tracing

- The software of **xfst** 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.

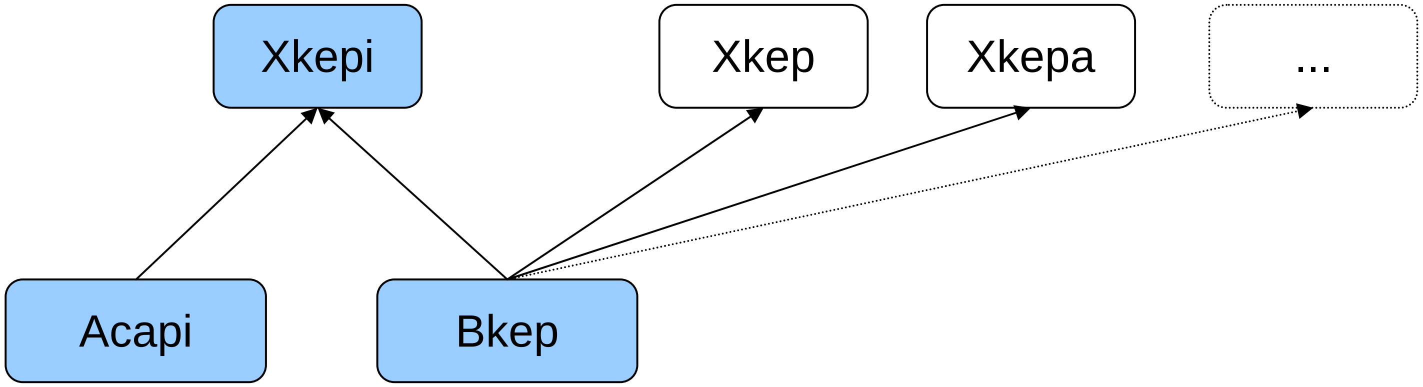
- Using the notation *R*.u to designate the upper language of a relation *R*, however, we simply need, in effect, to compute **apply up** for the individual daughter forms and then take the intersection of the results.

Our approach to upward tracing

Thus:

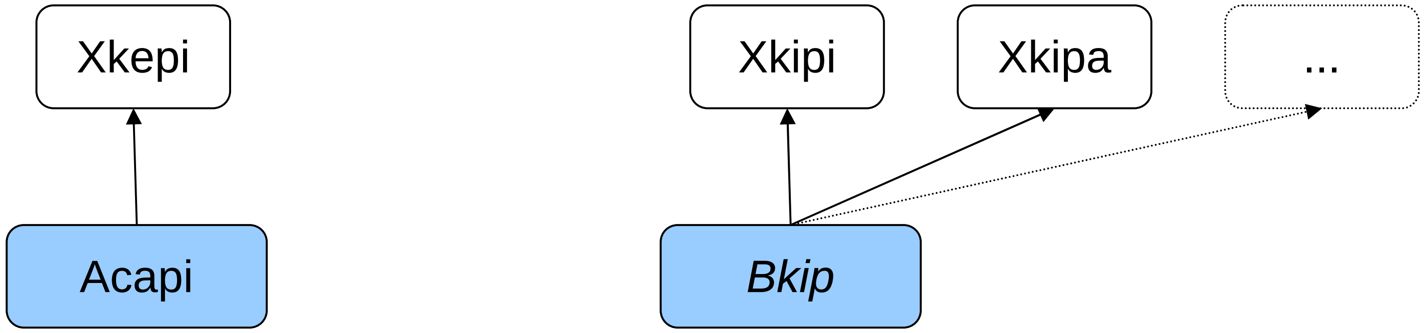
- **read regex [[HistPhon .o. {Acapi}].u & [HistPhon .o. {Bkep}].u] ;**
- **print words**

produce the string **Xkepi** as the unique possible antecedent.



Our approach to upward tracing

- On the other hand, if, e.g., **Bkip** is substituted for **Bkep** in this example, then no common antecedent can be computed for the given reconstruction.



Our approach to upward tracing

- The approach outlined here does not depend on a tree model of genetic relationship, although that has been chosen for the example. We can capture a wave model or network relationship by introducing a slightly modified technique.

Architecture of Cogs & Etyms: the language-description module

```
! toy-ex.txt

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

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

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

Architecture of Cogs & Etyms: the module for automatic writing of complicated regular expressions

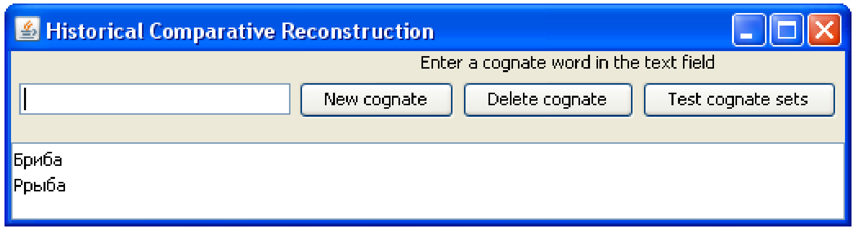
- The following needs to be typed in the command line of (x)fst/foma to test if **capi** from daughter language **A** and **kep** from daughter language **B** have a common etymon:

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

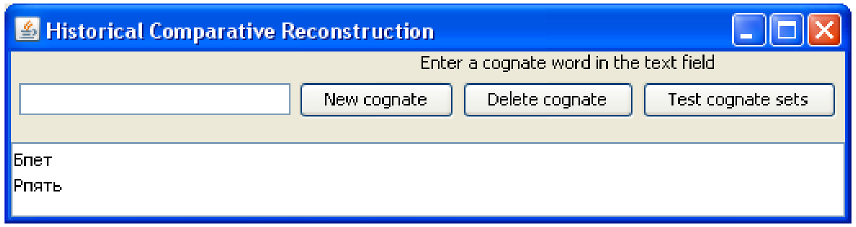
- Instead, our tool makes a call to a Perl script that dynamically creates an xfst/foma script that contains code similar to the lines above. The user only needs to type **Acapi** and **Bkep** in the Java GUI or the **name of the file** that contains the cognate sets.

Architecture of Cogs & Etyms: the Java GUI

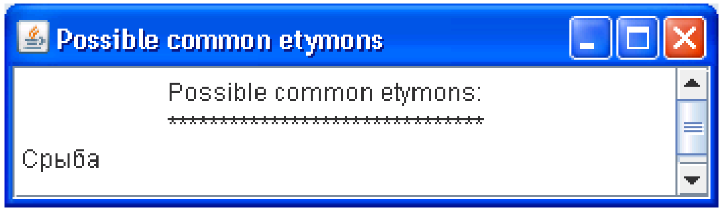
- We have built a Java GUI to facilitate the input and display of Unicode characters. Here is an example for Slavic (where ‘**Б**’, ‘**Р**’, ‘**С**’ are tags for Bulgarian, Russian, and Slavic, and ‘**E**’ transliterates ‘ѧ’):


Historical Comparative Reconstruction
Enter a cognate word in the text field
New cognate
Delete cognate
Test cognate sets

```
Бриба
Ррыба
```

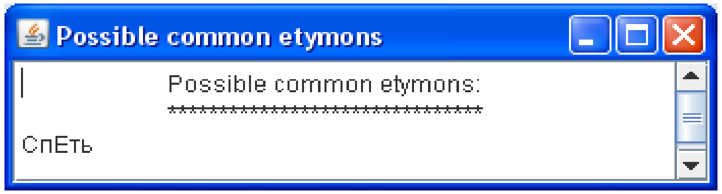

Historical Comparative Reconstruction
Enter a cognate word in the text field
New cognate
Delete cognate
Test cognate sets

```
Бпет
Рпять
```



```
Possible common etymons:
********************************

Срыба
```



```
Possible common etymons:
********************************

СпЕть
```

Architecture of Cogs & Etyms: input from a file

To test multiple cognate sets our application accepts input from a file:

Р:мясо	Б:месо
Р:день	Б:ден
Р:конь	Б:кон
Р:сон	Б:сън
Р:дуб	Б:дъб
Р:рыба	Б:риба
Р:снег	Б:сняг
Р:ветр	Б:вятър
…	…

Testing

Some reconstructed etymons for sets of Russian and Bulgarian cognate words:

Р:мясо	Б:месо	С:мЕсо	Е = ѧ
Р:день	Б:ден	С:дЬнЬ	Ь = ь
		С:дЯнЬ	Я = ѣ
		С:денЬ	
Р:конь	Б:кон	С:конЬ	
Р:сон	Б:сън	С:сЪнЪ	Ъ = ъ
Р:дуб	Б:дъб	С:дОбЪ	О = ѫ
Р:рыба	Б:риба	С:рыба	
Р:снег	Б:сняг	С:снЯгЪ	
Р:ветр	Б:вятър	С:вЯтрЪ	

Testing

- We have tested the technique with data from Slavic and Germanic languages.
- For Slavic we extracted automatically from the online version of Vasmer's Russian etymological dictionary over 1000 entries that contained both Bulgarian and Old Church Slavonic forms
- We selected over 700 pairs of Russian and Bulgarian words that formed cognate sets
- We reconstructed successfully Late Common Slavic etymons for each pair
- We compared our reconstructions with the OSC forms provided by Vasmer

Summary

Cogs & Etyms is a tool for upward tracing that:

- makes it possible to reconstruct common etymons from sets of cognate words from genetically related languages
- allows the researcher to identify deviations from the expected „correct“ daughter-language forms due to analogy, confusion, merger, etc.
- saves a lot of tedious, exhausting, boring and error-prone typing of programming code
- allows to test a large number of cognate sets

THANK YOU!