Interactive Etymological Inference via Statistical Relational Learning

Johannes Dellert

University of Tübingen

August 22, 2019

Plan

1. The Etymological Inference Engine
2. Use Case 1: Inferring Morphology
3. Use Case 2: Sound Correspondences and Loanword Detection
4. Prototype of the User Interface

Contents

The Etymological Inference Engine (EtInEn)

Purpose of the **Etymological Inference Engine (EtInEn)**:

- a computational system for historical linguistics which works and communicates with the user in classical terms, but is supported by a probabilistic model that is used to quantify strength of evidence
- linguist user can interact with the system by inserting and retrieving ideas of many types (cognacy judgments, reconstructions, soundlaws), but specialized reasoning components can also generate them without any user input
- system attempts to build a theory around the user's decisions, and notifies the user of inconsistencies in the current hypothesis
- final result is a consistent etymological theory in classical terms

EtInEn: General design principles

Main design principles of EtInEn:

- etymological theories are modeled and represented as **collections of atomic ideas** to which **belief values between 0 and 1** are assigned
- atoms are instances of a fixed set of **first-order predicates**, e.g. `Mspl(W,Stem,Suffix)` "`W` can be split into `Stem` and `Suffix`"
- all predicates to which belief is assigned **must be meaningful** and transparent to a historical linguist, e.g. `Ccog(deu:Gasse,eng:gate)` "German *Gasse* and English *gate* are cognates"
- reasoning results can be inspected in detail, every reasoning step is **explainable in classical terms** because the underlying predicates are
- the **user can manipulate the belief values** assigned to individual atoms a well as groups of atoms in order to express their intuitions, and to advance theory development

EtInEn: System architecture

Cornerstones of the technical implementation:

- large sets of atoms representing an etymological theory are maintained in a database for efficient retrieval
- probabilistic reasoning over these ground atoms is performed by means of large **graphical models**
- our choice of template language for these models: **Probabilistic Soft Logic (PSL)**, a language for statistical relational learning
- principle of **refinement cycles**: user can trigger re-inference runs of parts of the theory, which are performed in the background while the user can continue to inspect and revise other parts of the data
- different parts of the theory **interact by sharing sets of atoms** and their belief values

Probabilistic Soft Logic (PSL)

Capabilities of PSL:

- support for **arithmetic rules** to reason directly over belief values (e.g. constraining sets of atoms to form a probability distribution)
- support for **disjunctive logical rules**, with semantics defined by Łukasiewicz t-co-norm $x_1 \vee x_2 := \min\{x_1 + x_2, 1\}$; this part could be described as weighted logic programming
- both types of rules can be **constraints** (which are never violated if at all possible), or **rules with learnable weights** (i.e. to empirically determine weights for different types of evidence)
- efficient inference optimizes belief values in such a way that the **distance to satisfaction** ("pressure") over all grounded rules is minimized

Implementing etymological reasoning in PSL

Applying PSL to etymological reasoning:

- constraints are used for enforcing the relevant **principles** of the comparative method (e.g. consistency of reconstructions)
- weighted rules and priors are used to model the less precise **heuristics** commonly employed in the field, such as intuitions about which sound changes or semantic shifts are plausible
- rules are written with variables, the **actual inference works on all possible groundings** of these rules (i.e. implicit universal quantification)
- existing reasoning components, e.g. for cognate detection, function as **idea generators**, i.e. they determine which grounded atoms to inject into the database, steering which groundings enter the computation

State of EtInEn

Current situation:

- finished implementing a lot of supporting technology around PSL
- first versions of PSL models for central reasoning components exist (bulk of this talk)
- prototype of the user interface exists (end of the talk)
- graphical components for inspecting and interacting with central parts of the etymological theory exist at various stages of completion

Next steps:

- integrating the partial models into full theory modeling
- compact storage of the current system state
- training rule weights and evaluation based on etymological data

Contents

Inferring Morphology: Test Data

Turkish as a test case for simple stem-suffix splitting:

- agglutinative language with a lot of derivational morphology
- advantage: morpheme boundaries are generally clear-cut
- difficulty: vowel and consonant harmony

Test data for morphological inference:

- list of 1,249 Turkish lemmas, assigned to 1,016 concepts and automatically transcribed into IPA (from NorthEuraLex database)
- frequent derivational patterns between the concepts compiled into a weighted loose colexification network (modeling information that e.g. FIRST is frequently derived from ONE, FEED from EAT, etc.)

Inferring Morphology: Model

Vocabulary of predicates for talking about morphology:

- **`Mfre(Morpheme)`**: `Morpheme` occurs as a free morpheme
- **`Mbnd(Morpheme)`**: `Morpheme` occurs as a bound morpheme
- **`Mspl(Word,Stem,Suffix)`**: `Word` is derived from `Stem` by `Suffix`

Helper predicates for modeling the input data:

- **`Xsem(Word,Concept)`**: `Word` has meaning `Concept`
- **`Xlen(Morpheme,Length)`**: `Morpheme` is of length `Length`
- **`Xspl(Word,Stem,Suffix)`**: `Word` could consist of `Stem` and `Suffix`
- **`Lwcl(Concept1,Concept2)`**: colexification weight between Concept1 and Concept2 (derived from number of derivations in database)

Inferring Morphology: Model

Core rules of the PSL model:

- `Mspl(Word, +Stem, +Suffix) = 1 .`
- `Mspl(Word, Stem, Suffix) -> Mbnd(Suffix) .`
- `Mspl(Word, Stem, Suffix) -> Mfre(Stem) .`
- `1: Mfre(Stem) & Xspl(Word, Stem, '-') -> Mspl(Word, Stem, '-')`
- `W1 != W2 & Xsem(W1,C1) & Xsem(W2,C2) & Lwcl(C1,C2) & Mspl(W1, R, S1) & Xspl(W2, R, S2) -> Mspl(W2, R, S2) .`

Additional rules:

- weighted rules of shape `Xlen(M,i) -> ~Mbnd(M)`
  for encoding priors over the expected length of bound morphemes
- rules for encoding priors over expected length of free morphemes:
  `20: Xlen(M,'1') -> ~Mfre(M)`
  `5: Xlen(M,'2') -> ~Mfre(M)`

Inferring Morphology: Model

Idea generator for morphological inference:

- needs to decide which splits are considered by the model
- this is done by committing `Xspl(Word,Stem,Suffix)` atoms to the underlying database (which will determine the groundings)
- current procedure for generating `Xspl` atoms:
  - maintain a count of suffixes occurring in the data
  - only generate splits `Xspl(Word, Stem, Suffix)` where the `Suffix` was seen at least twice in the data
  - also generate the trivial split `Xspl(Word, Stem, '-')` for every word

Inferring Morphology: Results

92%	-dan	bura**dan**, sonra**dan**	++ (case ending)
88%	-lik	zengin**lik**, gerçek**lik**, pis**lik**, ...	++
86%	-k	(some belief for hundreds of words)	– –
84%	-lık	sıcak**lık**, hasta**lık**, ...	++ (cf. -lik)
84%	-nci	iki**nci**	++
81%	-mak	vur**mak**, anla**mak**, dur**mak**, ...	++
76%	-luk	uzun**luk**, doğru**luk**, soğuk**luk**, ...	++ (cf. -lik)
76%	-lemek	temiz**lemek**, ...	+
76%	-lamak	su**lamak**, baş**lamak**, ..	+ (cf. -lemek)
74%	-landırmak	ad**landırmak**, duygu**landırmak**	+
71%	-n	o**n**, yakı**n**, uzu**n**, ...	– –
71%	-li	keder**li**, neşe**li**, ...	++
66%	-ımak	taş**ımak**	++
66%	-inci	bi**rinci**	++ (cf. -nci)
66%	-nlar	o**nlar**, insa**nlar**	– (-lar is plural)
65%	-rada	bu**rada**, şu**rada**, o**rada**	+
63%	-mek	git**mek**, ver**mek**, iste**mek**, ...	++ (cf. -mak)

Contents

Loanword Detection: Test Data

First experiment in loanword detection:

- take a pair of distantly related languages which share a common layer of loans from a third language (or group of languages)
- attempt to use sound correspondences to classify words which appear etymologically related as either due to shared inheritance or borrowing

Test case:

- Finnish and Hungarian, from different branches of Uralic
- age of latest common ancestor: at least 4000 years
- about 200 items of shared basic vocabulary
- both influenced by neighboring Indo-European languages, resulting in quite a few shared loans
- evaluation on 100 word pairs with largest surface similarity

Loanword Detection: Test Data (Extract)

tee	TEA	tea
sarvi	HORN	szarv
me	WE	mi
te	YOU	ti
voi	BUTTER	vaj
mestari	MASTER	mester
veri	BLOOD	vér
öljy	OIL	olaj
syy	SIN	bűn
heinä	HAY	széna
koputtaa	KNOCK	kopogtat
levy	PLATE	lemez
laji	SPECIES	faj
uusi	NEW	új
kutoa	KNIT	köt
vesi	WATER	víz
varis	CROW	varjú
nuora	STRING	zsinór

koira	DOG	kutya
ajaa	DRIVE	jár
valita	SELECT	választ
kukko	ROOSTER	kakas
puoti	SHOP	bolt
mänty	PINE	fenyő
niellä	SWALLOW	nyel
risti	CROSS	kereszt
käsi	HAND	kéz
kyynel	TEAR	könny
levy	SHEET	lap
pesä	NEST	fészek
lapio	SHOVEL	lapát
kala	FISH	hal
pää	HEAD	fő
riemu	JOY	öröm
kivi	STONE	kő
petäjä	PINE	fenyő

Loanword Detection: Test Data (Categorized)

tee	TEA	tea
sarvi	HORN	szarv
me	WE	mi
te	YOU	ti
voi	BUTTER	vaj
mestari	MASTER	mester
veri	BLOOD	vér
öljy	OIL	olaj
syy	SIN	bűn
heinä	HAY	széna
koputtaa	KNOCK	kopogtat
levy	PLATE	lemez
laji	SPECIES	faj
uusi	NEW	új
kutoa	KNIT	köt
vesi	WATER	víz
varis	CROW	varjú
nuora	STRING	zsinór

koira	DOG	kutya
ajaa	DRIVE	jár
valita	SELECT	választ
kukko	ROOSTER	kakas
puoti	SHOP	bolt
mänty	PINE	fenyő
niellä	SWALLOW	nyel
risti	CROSS	kereszt
käsi	HAND	kéz
kyynel	TEAR	könny
levy	SHEET	lap
pesä	NEST	fészek
lapio	SHOVEL	lapát
kala	FISH	hal
pää	HEAD	fő
riemu	JOY	öröm
kivi	STONE	kő
petäjä	PINE	fenyő

Loanword Detection: Sound Correspondences

Manually modeled single-segment correspondences, compared to EtInEn suggestions:

fin	hun	Comment	Found
/k/	/k/	# _ [+front]	yes
/t/	/t/		yes
/p/	/p/	pp ~ p	no
/s/	/s/	PU /ś/	yes
/r/	/r/		yes
/l/	/l/		yes
/n/	/n/		yes
/j/	/j/		yes
/ʊ/	/v/		yes
/ɛ/	/e/		yes
/i/	/ɛ/		yes
/ɑ/	/a/		yes

fin	hun	Comment	Found
/k/	/h/	# _ [+back]	yes
/t/	/d/	nt ~ d	yes
/p/	/f/	# _	yes
/p/	/v/	V _ V	no
/n/	/ɲ/		yes
/m/	/v/	V _ #	no
/æ/	/e/		yes
/æ/	/g/	[+length] ~g	yes
/y/	/g/	[+length] ~g	no
/i/	/g/	[+length] ~g	no
/u/	/g/	[+length] ~g	no
/e/	/g/	[+length] ~g	no

Loanword Detection: Model

Vocabulary of predicates for talking about cognacy and loanwords:

- **`Chom(Word1, Word2)`**: the words are etymologically related
- **`Ccog(Word1, Word2)`**: the words are cognate
- **`Cloa(Word1, Word2)`**: the words are shared loans

Helper predicates for modeling the input data:

- **`Wsim(Word1, Word2)`**: overall sequence similarity
- **`Ssim(Sound1, Sound2)`**: global sound similarity
- **`Scor(Sound1, Sound2)`**: sound correspondence [0 or 1]
- **`Aali(Word1, Word2, Pos, Sound1, Sound2)`**: in the pairwise alignment of words `Word1` and `Word2`, `Sound1` and `Sound2` are aligned at position `Pos`

Loanword Detection: Model

Core rules of the PSL model:

- `Ccog(W1,W2) + Cloa(W1,W2) = Chom(W1,W2) .`
- `Wsim(W1, W2) -> Chom(W1, W2) .`
- `1: Aali(W1, W2, Pos, S1, S2) & Ssim(S1, S2) -> Cloa(W1, W2)`
- `Aali(W1, W2, Pos, S1, S2) & Chom(W1, W2) & ~ Scor(S1, S2) & Ssim(S1, S2) -> Cloa(W1, W2) .`
- `1: Aali(W1, W2, Pos, S1, S2) & ~ Ssim(S1, S2) & Scor(S1, S2) -> Ccog(W1, W2)`

Negative priors, with weights decided based on development set of 15 word pairs:

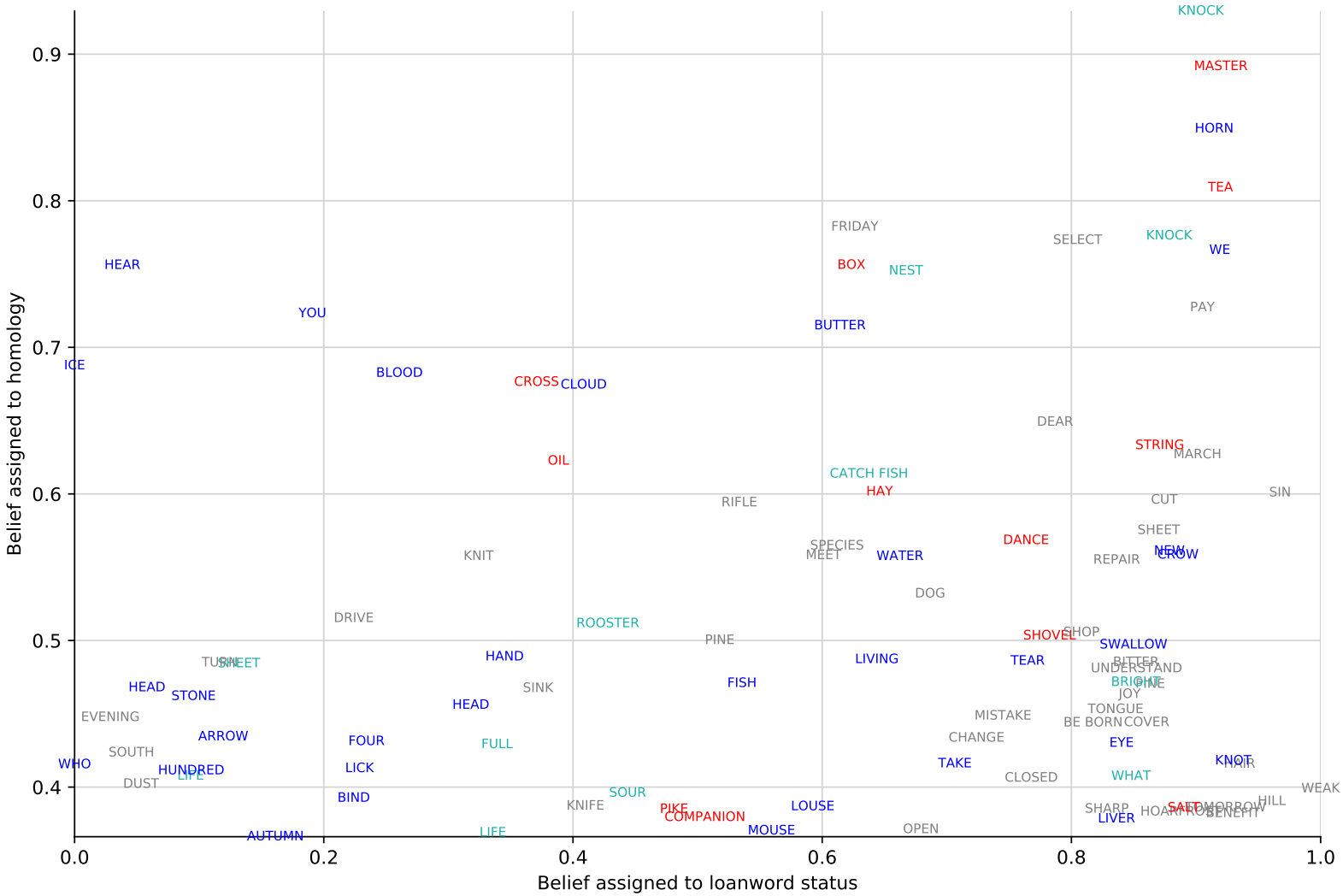
- `1: ~Chom(W1, W2)`
- `3: ~Cloa(W1, W2)`

Loanword Detection: Model

Idea generation mostly means data import and preprocessing:

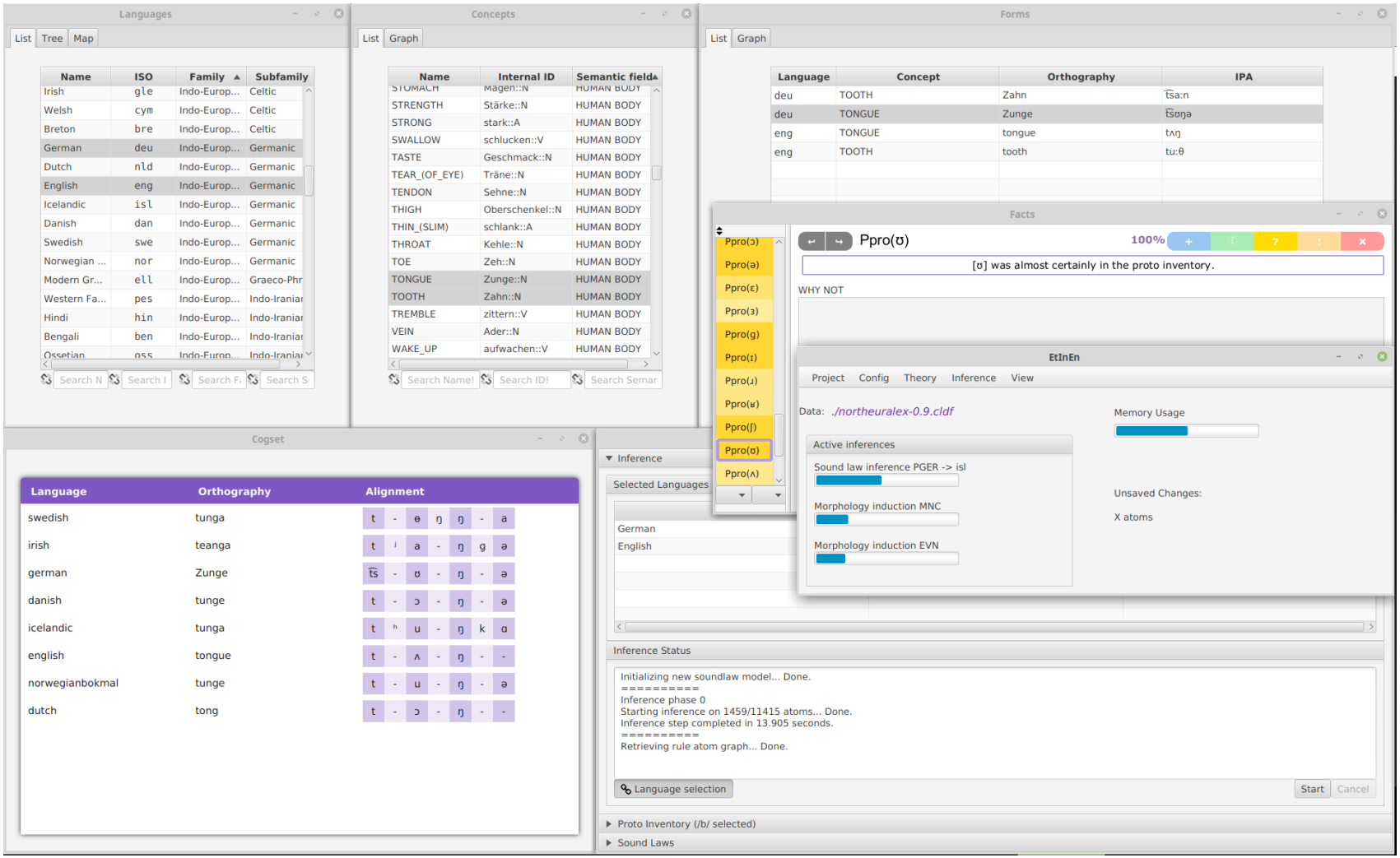
- perform information-weighted sequence alignment (IWSA) with global weights on each word pair (for more details, see Dellert 2018)
- use output of IWSA to generate `Aali` and `Wsim` atoms
- import `Ssim` values from global sound similarity matrix (also inferred using IWSA on the entire NorthEuraLex database)

Loanword Detection: Result



Contents

EtInEn User Interface: Current State



EtInEn User Interface: Components

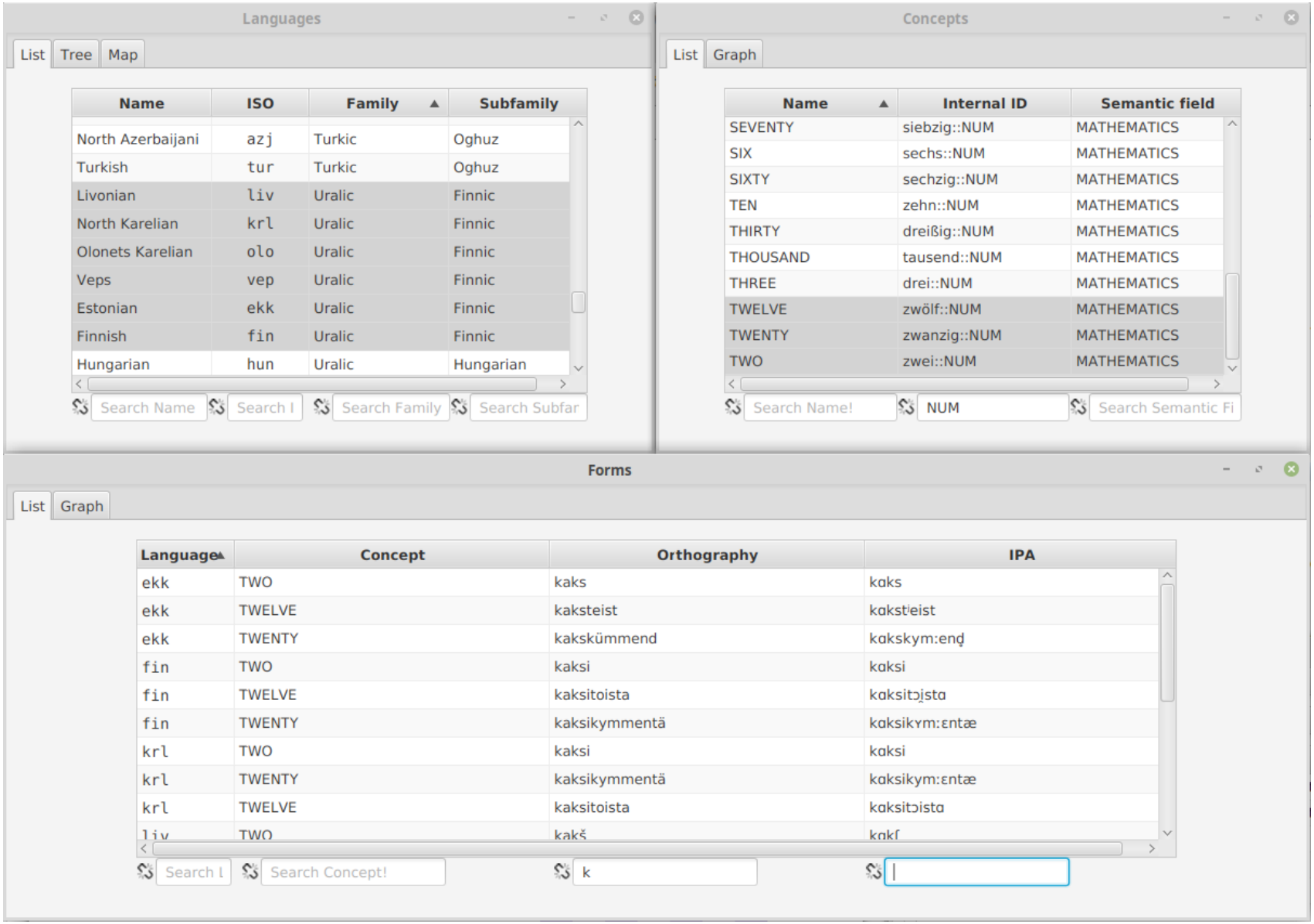
Basic facts about EtInEn interface:

- written entirely in Java (i.e. a stable platform)
- standalone Desktop application using JavaFX
- consists of many windows which can be distributed and freely arranged across several monitors

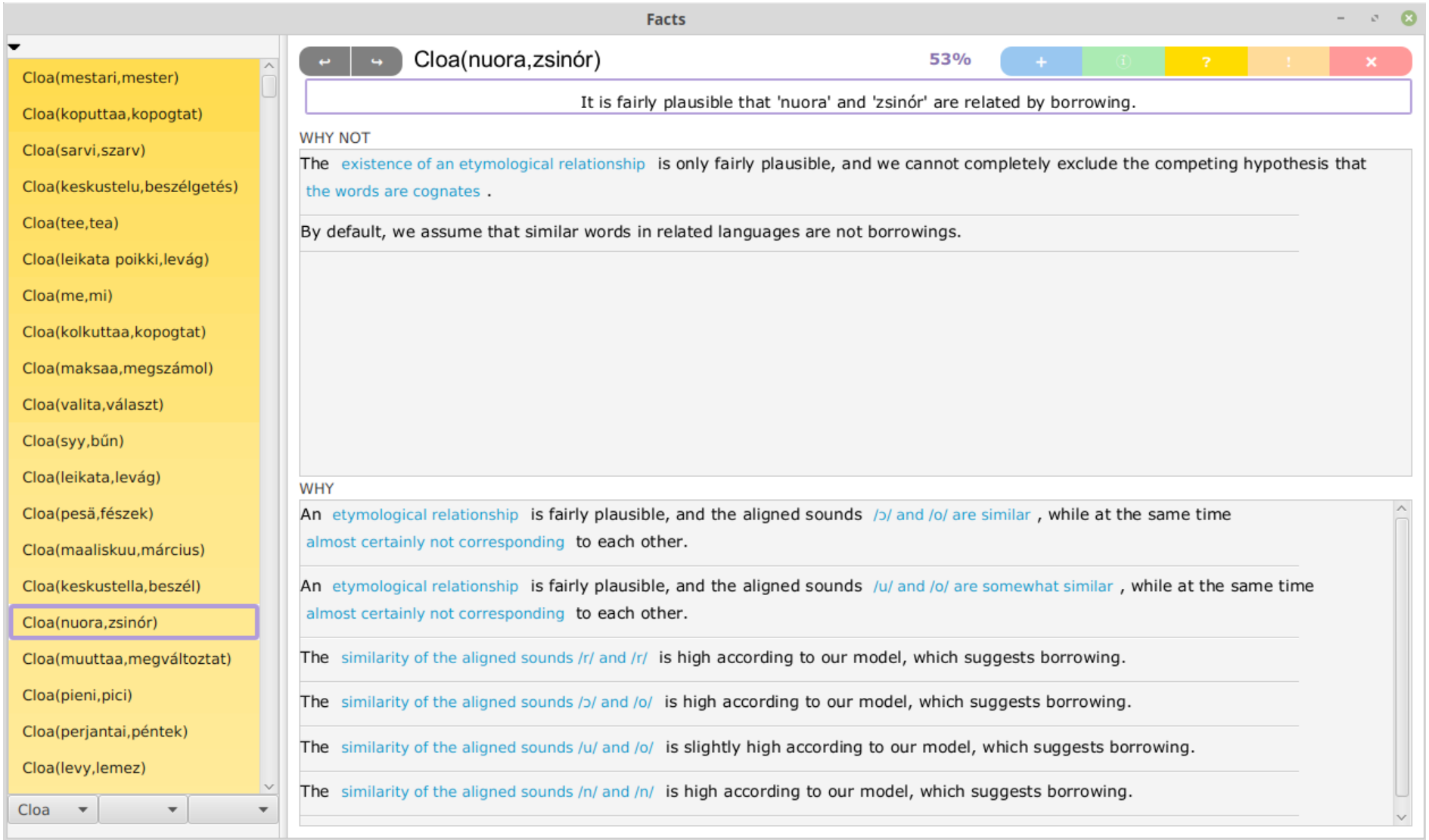
Currently implemented components:

- list-based selection windows for languages, concepts, forms
- cognate set inspection with alignment visualization
- interactive component for soundlaw inference
- fact explorer for inspecting (and manipulating) atoms and their connections (also used as a standalone debugging tool for testing and debugging PSL models)

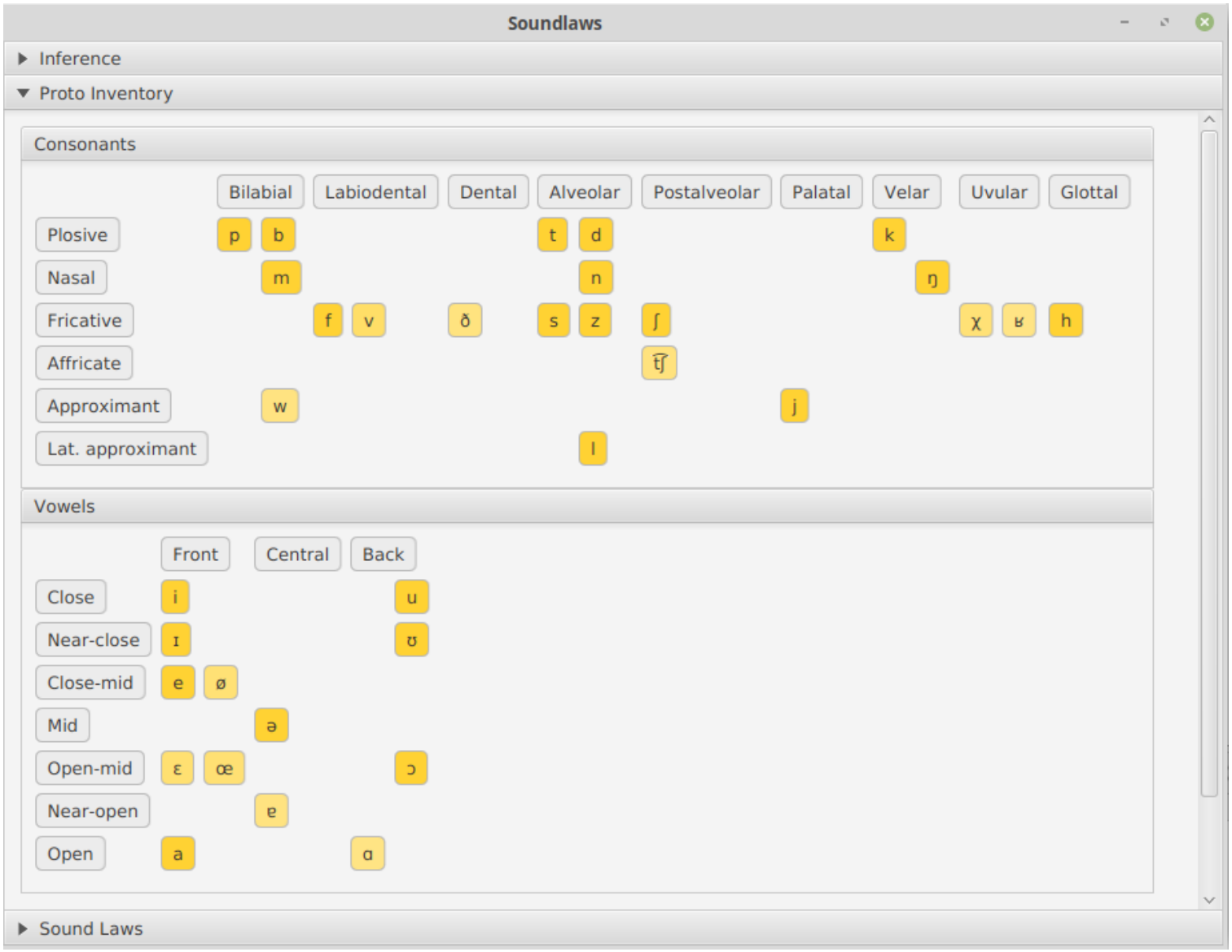
EtInEn User Interface: Selecting Data



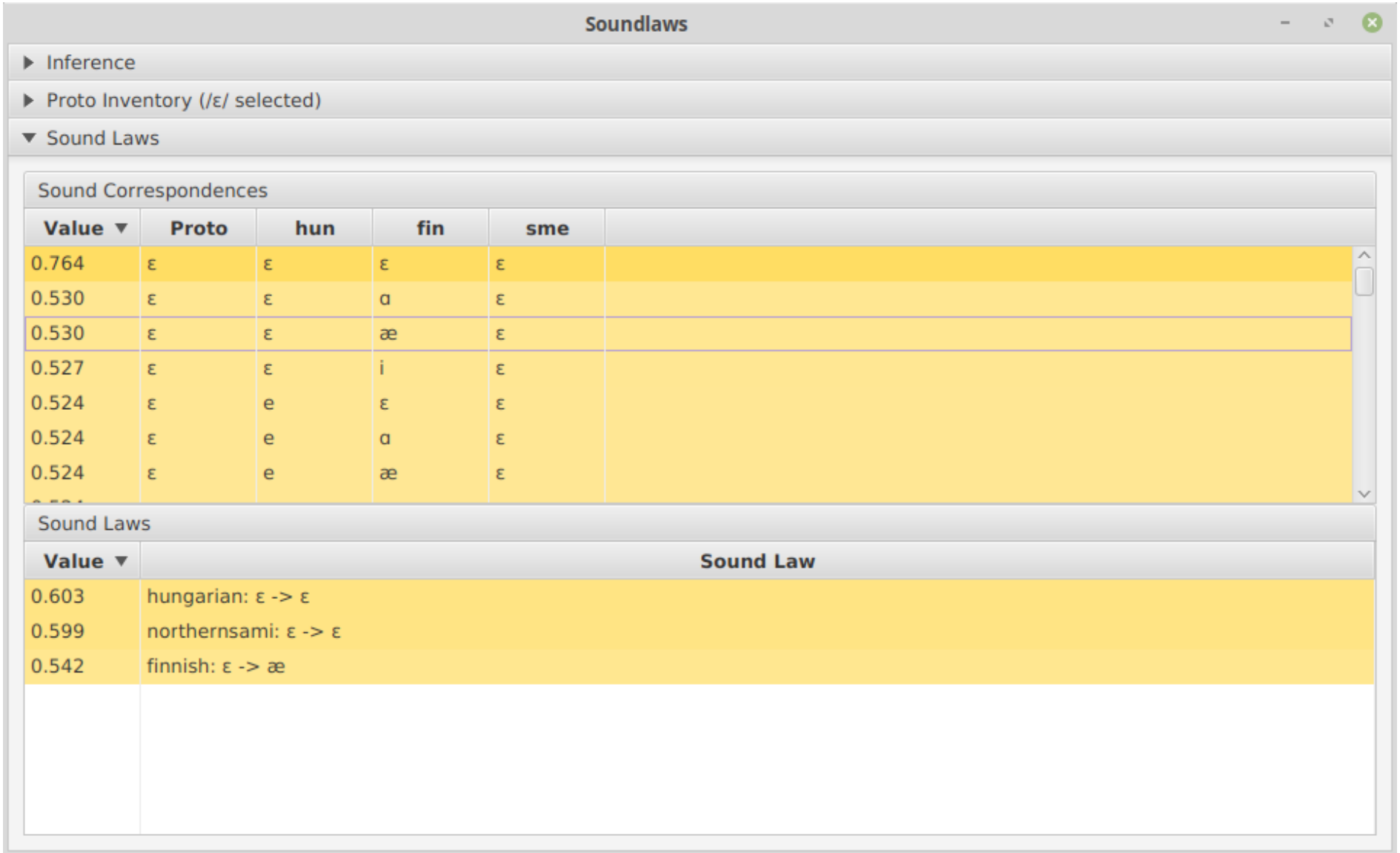
EtInEn User Interface: Inspecting Inference Results



EtInEn User Interface: Proto-Inventories



EtInEn User Interface: Sound Laws



Acknowledgments: My Team

I would like to thank the students on my team for their contributions:

- Thora Daneyko (PSL infrastructure and sound correspondences)
- Jekaterina Kaparina (user interface and CLDF import code)
- Zhuge Gao (morphology detection)
- Verena Blaschke (loanword detection, verbalizing results)
- Rahel Albicker, Maxim Korniyenko, Anna Karnysheva, Zhuge Gao, Yuliya Mkhayan, Anastasia Buianova (lexical data collection)
- Anna Bródy, Karina Hensel, Živilė Rasimaitė, Rahel Albicker (etymological annotation)

Acknowledgments: Funding

Work on this project has been funded by:

- DFG Center for Advanced Studies "Words, Bones, Genes, Tools: Tracking Linguistic, Cultural and Biological Trajectories of the Human Past" (half-time position as a fellow)
- the intramural Program for the Promotion of Junior Researchers, part of the Institutional Strategy of the University of Tübingen, DFG ZUK 63 (data collection and annotation)
- RiSC grant from the Baden-Württemberg Ministry of Education (programmers, additional funds for data collection)

Thank you for your attention!

References

Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-Loss Markov Random Fields and Probabilistic Soft Logic. *Journal of Machine Learning Research (JMLR)*, 2017.

Johannes Dellert. Combining Information-Weighted Sequence Alignment and Sound Correspondence Models for Improved Cognate Detection. 2018. 27th International Conference on Computational Linguistics (COLING 2018).

Johannes Dellert, Thora Daneyko, Alla Münch, Alina Ladygina, Armin Buch, Natalie Clarius, Ilja Grigorjew, Mohamed Balabel, Isabella Boga, Zalina Baysarova, Roland Mühlenbernd, Johannes Wahle, and Gerhard Jäger. NorthEuraLex: A Deep-Coverage Lexical Database of Northern Eurasia. Under revision at Language Resources and Evaluation, 2018.

Approaches to Historical Linguistics

Classical methods:

- take all the available evidence into account
- ideally results in consistent theories which explain large parts of each language's lexicon and grammar
- not fully formalizable
- problems if there is conflicting evidence
- many interesting questions (e.g. dating) beyond scope

Computational methods:

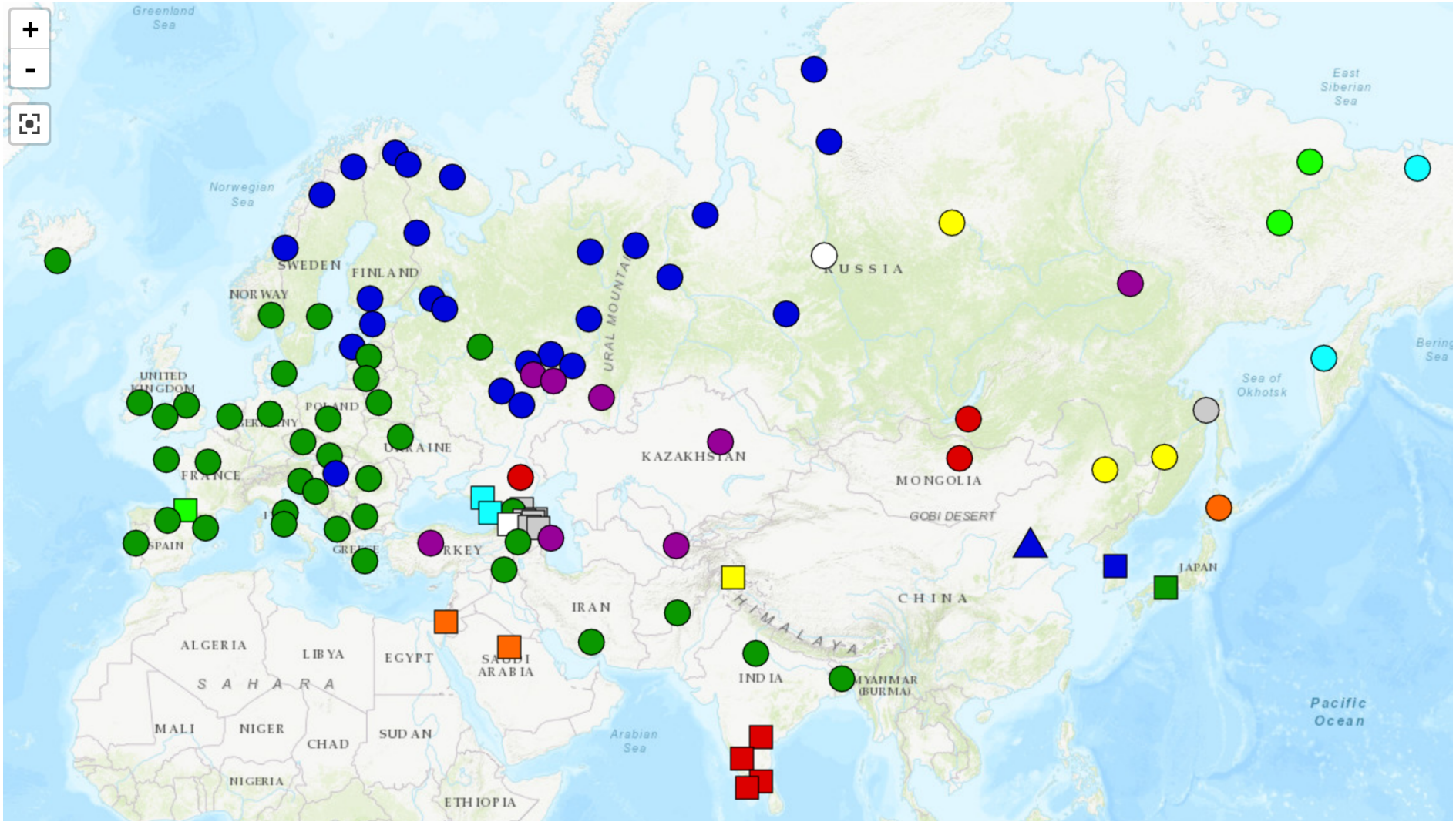
- work with a small, carefully sampled subset of the data
- ideally help to decide long-standing open questions by providing a framework for dealing with uncertainty
- based on mathematical models of evolution
- evidence is quantifiable, but difficult to interpret
- studies often contradictory

NorthEuraLex

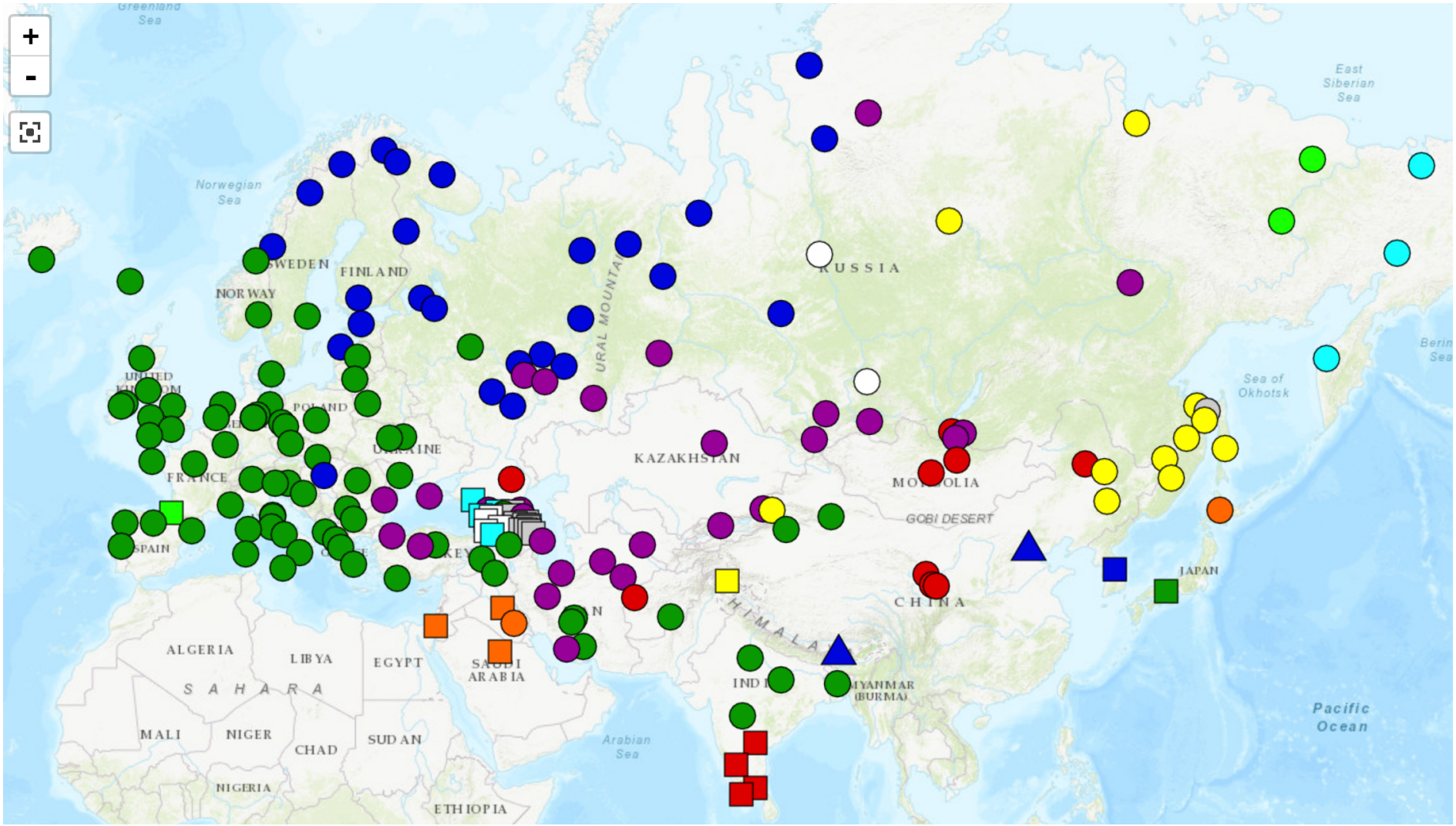
Scope of the **NorthEuraLex** database:

- list of 1,016 cross-linguistically applicable concepts
- collect realizations of these concepts across all sufficiently documented languages of Northern Eurasia (107 languages from 20 different families at the moment, currently expanding to 196 languages)
- (mostly) automated transcription of collected words into the International Phonetic Alphabet to make the data comparable
- etymological annotation (full and partial cognacy, loanwords) for well-researched language families is under way

NorthEuraLex 0.9 (situation in June 2018)



NorthEuraLex 1.0 (projected situation in June 2020)



Etymological Annotation

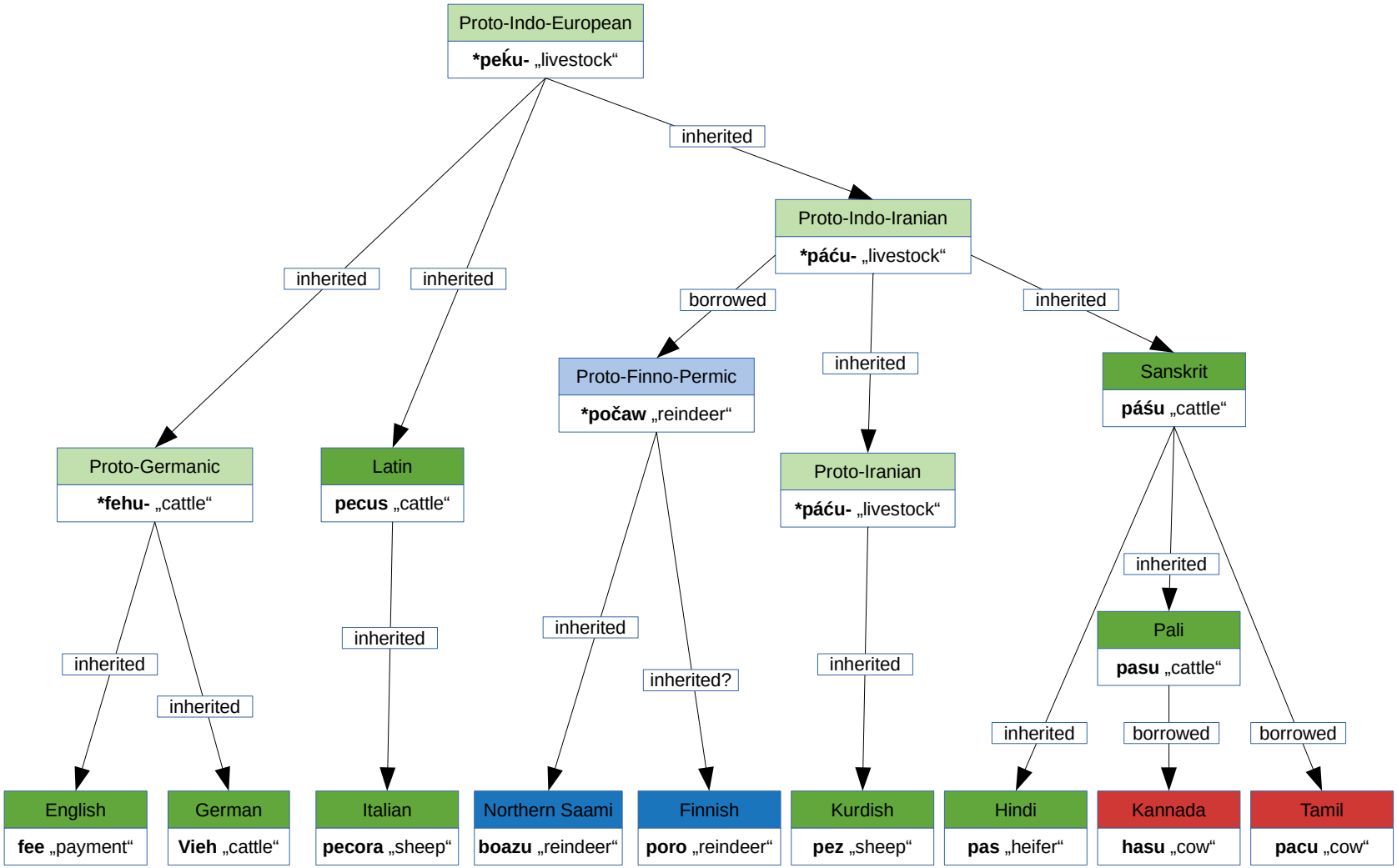
Challenges of etymology modeling:

- etymologies are not data, but theories (people disagree!)
- we are not experts on any language family, i.e. we can't make decisions, and strive to faithfully model the experts' opinions
- words are quoted in widely divergent shapes by different sources

Current state of etymological annotation:

- data format is final, annotation manual almost finished
- complex toolbox implementing various processing steps
- problem: many sources do not cover all languages in a (sub)family

Etymological Annotation: Example



PSL: Terms

The basic definitions are well-known from logic programming:

- a **term** is either a constant or a variable
- a **constant** is a unique string that denotes an element in the universe over which the program is grounded
- a **variable** is an identifier for which constants can be substituted

In the PSL reference implementation,

- constants must be delimited by double or single quotes
- identifiers start with a letter, and can contain digits and underscores (by convention: first letter uppercase)

PSL: Predicates and Atoms

Predicates encode relations between terms:

- a **predicate** is a relation defined by a unique identifier and an arity (number of arguments)
- an **atom** is a predicate combined with a sequence of terms of length equal to the predicate's arity
- a **ground atom** is an atom with only constants for arguments
- PSL reasons over probabilities assigned to each ground atom in a pre-defined universe
- variables in the specification are merely placeholders for compactly defining the **interactions between ground atoms**

PSL: Input Format

A data set is represented in the form of four objects:

- a set $\mathbb{C}$ of closed predicates with completely observed atoms
- a set $\mathbb{O}$ of open predicates whose atoms may be unobserved
- a base $\mathcal{A}$ of all atoms under consideration
- a function $\mathcal{O}\colon \mathcal{A} \to [0,1] \cup \{\emptyset\}$ mapping ground atoms to either an observed value or unobserved state

PSL: Typed Constants

In the input format, the base can be defined implicitly:

- constants in the universe are grouped into types

```
Person = {"alexis", "bob", "claudia", "david"}
Professor = {"alexis", "bob"}
Student = {"claudia", "david"}
Subject = {"computer science", "statistics"}
```

- arguments in predicate declaration constrain the constants they will be grounded over; annotation `closed` marks predicates in $\mathbb{C}$

```
Advises(Professor, Student)
Department(Person, Subject) (closed)
```

PSL: Logical Rules

- a **literal** is an atom or a negated atom (notation: `!` or `_`)
- a **logical rule** is a disjunctive clause of literals
- this includes **implications** (written `->`) with a conjunction of literals (`&`) in the body and a disjunction (`|`) in the head
- typical form of a logical rule:
  `P1(A,B) & P2(A,B) -> P3(A,B) | P4(A,B)`
- helper predicates are needed to enforce a cluster of jointly true atoms in the consequent (this is crucial to keep things tractable)

PSL: Weighted Rules and Constraints

Rules in PSL can be weighted or unweighted:

- a **weighted rule** quantifies the degree to which non-satisfaction will be penalized, notation uses a non-negative weight as prefix:
  `1 : Advisor(Prof, S) & Dep(Prof, Sub) -> Dep(S, Sub)`
- an unweighted rule or **constraint** requires the rule to always be satisfied, notation uses a dot at the end:
  `Parent(A,B) & Parent(B,C) -> Grandparent(A,C) .`
- this allows a mixture of hard constraints and statistical modeling
- using weighted rules of a single disjunct, we can define **priors**
- rule weights can be learned
  (PSL is a tool for **statistical relational learning**)

PSL: Arithmetic Rules

The second rule type directly reasons with the values of atoms:

- a **summation atom** represents the sum over all constants for sum variables marked by +:
  `Friends("John",+Friend)` is the sum over all atoms of shape `Friends("John",Friend)`, i.e. it counts the number of friends
- an **arithmetic rule** relates two linear combinations of (summation) atoms with an inequality or equality
- Example: mutual exclusivity of being liberal or conservative
  `Liberal(P) + Conservative(P) = 1 .`
- Example: we strongly prefer each token to have a single category:
  `100 : HasPOS(Token,+POS) = 1`

Semantics of Connectives

Semantics of connectives between "belief scores" taken from fuzzy logic:

- negation is simply the belief assigned to the opposite: $\neg x := 1 - x$
- **conjunction** is implemented by the Lukasiewicz t-norm: $x_1 \wedge x_2 := \max\{x_1 + x_2 - 1, 0\}$
- **disjunction** is implemented by the Lukasiewicz t-co-norm: $x_1 \vee x_2 := \min\{x_1 + x_2, 1\}$

For values 0 and 1, behaviour is equivalent to Boolean logic.

Distance to Satisfaction

- let $\mathbf{y} = (y_1, ..., y_n)$ and $\mathbf{x} = (x_1, ..., x_{n'})$ be vectors of variables with joint domain $D = [0, 1]^{n+n'}$, where the $\mathbf{y}$ are associated with grounded atoms over open predicates, and $\mathbf{x}$ with closed predicates
- **distance to satisfaction** of a disjunctive clause of positive atoms with indices I_j^+ and negated atoms with indices I_j^-:

$$\min\{\sum_{i \in I_j^+} y_i + \sum_{i \in I_j^-} (1 - y_i), 1\}$$

- intuitively: quantification of the unsatisfiedness of an implication
- distances to satisfaction are linear combinations to minimize
- constraints are linear equations forcing some distances to be 0
- more general arithmetic constraints are a natural extension

Hinge-Loss Energy Function

- let $\phi = (\phi_1, \ldots, \phi_m)$ be a vector of continuous potentials of the form $\phi_j(\mathbf{y}, \mathbf{x}) = (\max\{l_j(\mathbf{y}, \mathbf{x}), 0\})^{p_j}$, where each l_j is a linear function of $\mathbf{y}$ and $\mathbf{x}$, and $p_j \in \{1, 2\}$
- given a vector of non-negative weights $\mathbf{w} = (w_1, \ldots, w_m)$, a **hinge-loss energy function** $f_{\mathbf{w}}$ is then defined by

$$f_{\mathbf{w}} = \sum_{j=1}^{m} w_j \phi_j(\mathbf{y}, \mathbf{x}) \tag{1}$$

- intuitively: weighted sum of distances to satisfaction

Hinge-Loss Markov Random Fields

A PSL program induces a special class of graphical model:

- for linear constraint functions $\mathbf{c} := (c_1, ..., c_r)$ split into equality constraints $\mathcal{E}$ and inequality constraints $\mathcal{I}$, the **feasible subset** $\tilde{D}$ of the joint domain D is defined as

$$\tilde{D} := \left\{ (\mathbf{y}, \mathbf{x}) \in D \;\middle|\; \begin{array}{ll} c_k(\mathbf{y}, \mathbf{x}) = 0 & \forall k \in \mathcal{E} \\ c_k(\mathbf{y}, \mathbf{x}) \leq 0 & \forall k \in \mathcal{I} \end{array} \right\}.$$

- a **hinge-loss Markov random field (HL-MRF)** P over $\mathbf{y}$ conditioned on $\mathbf{x}$ is a probability density defined as $P(\mathbf{y}|\mathbf{x}) := 0$ for $(\mathbf{y}, \mathbf{x}) \notin \tilde{D}$, and as follows for $(\mathbf{y}, \mathbf{x}) \in \tilde{D}$:

$$P(\mathbf{y}|\mathbf{x}) := \frac{1}{Z(\mathbf{w}, \mathbf{x})} \exp(-f_w(\mathbf{y}, \mathbf{x}))$$

MAP inference in PSL

- Finding the maximum a-posteriori (MAP) estimate of belief assigned to facts amounts to minimizing hinge-loss:

$$\arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \equiv \arg\min_{\mathbf{y}|\mathbf{y},\mathbf{x}\in\tilde{D}} f_w(\mathbf{y},\mathbf{x})$$

- this can be done efficiently via the alternating direction of multipliers (ADMM) method on a consensus optimization reformulation
- further speedup by lazy MAP inference:
  if a feasible assignment minimizes the sum over a subset of the potentials, and all other potentials are 0 at this assignment,
  we can be sure to have found a MAP state

PSL: The Software

The reference implementation of PSL is

- available at `https://github.com/linqs/psl`
- open-source under an Apache license
- written in Java (easy interfacing)
- designed for and capable of parallel processing
- already used in many large-scale applications
  (e.g. social network analytics, topic modeling)
- proven to be much faster than general-purpose optimization on complex problems (like record unification or causal inference)