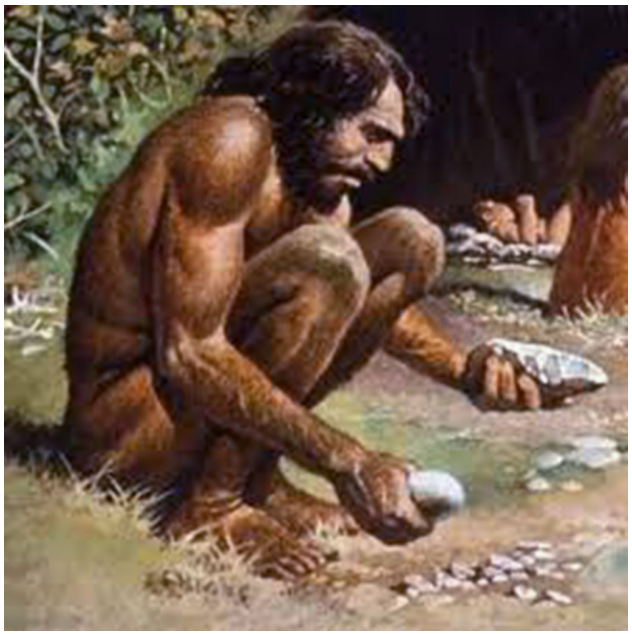
What can we learn from implementing Optimality Theory?

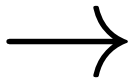
Tamás Biró

ELTE Eötvös Loránd University

PTA workshop @ GLOW 41, Budapest, April 14, 2018

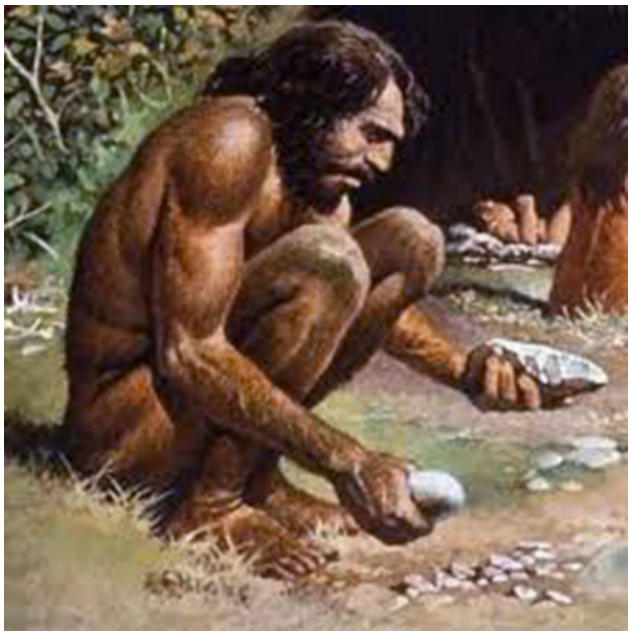
From a tool to insights and a novel perspective





$+, -, \times, \div \sqrt{}$ $\rightarrow$ $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

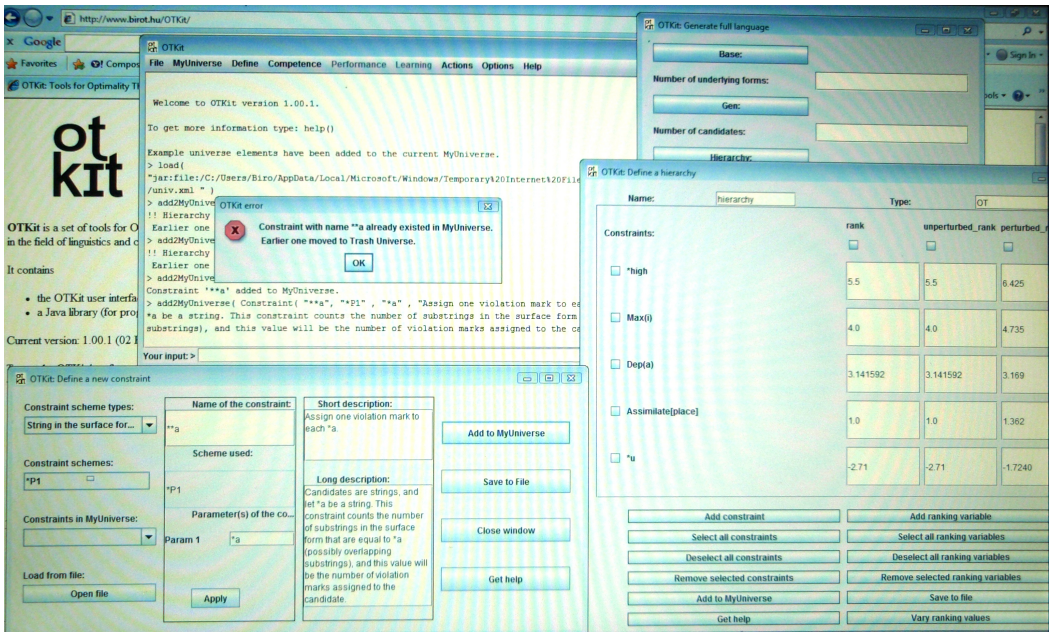
From a tool to insights and a novel perspective

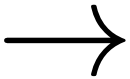


$\rightarrow$

$+, -, \times, \div \sqrt{}$ $\rightarrow$ $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

From a tool to insights and a novel perspective





Overview

1. Tools for Optimality Theory
2. Formalizing Optimality Theory
3. The fertilizing effect of looking at Gestalt pictures

Overview

The insufficiency of paper-and-pencil OT

Lauri Karttunen (2006). 'The insufficiency of paper-and-pencil linguistics: the case of Finnish prosody'. (ROA-818.)

> *"Without a* GEN *function to enumerate all the possible outputs, it is easy to miss the actual winner even if one is a native speaker of the language and an expert in the field."*

> *"Quandoque bonus dormitat Homerus."* *[Even good old Homer nods.]*

The linguist constantly feeling some insecurity:

'Have I not made a mistake in my analysis? Are all relevant candidates included? Have I correctly listed the violation marks in the tableau?'

The insufficiency of paper-and-pencil OT

Lauri Karttunen (2006). 'The insufficiency of paper-and-pencil linguistics: the case of Finnish prosody'. (ROA-818.)

> *"Without a* GEN *function to enumerate all the possible outputs, it is easy to miss the actual winner even if one is a native speaker of the language and an expert in the field."*

> *"Quandoque bonus dormitat Homerus."* *[Even good old Homer nods.]*

The linguist constantly feeling some insecurity:

'Have I not made a mistake in my analysis? Are all relevant candidates included? Have I correctly listed the violation marks in the tableau?'

Software tools for Optimality Theory

Available tools supporting the linguist include:

- OTSoft (Bruce Hayes, with contributions by Bruce Tesar and Kie Zuraw)
  `http://linguistics.ucla.edu/people/hayes/otsoft/`
- PRAAT (Paul Boersma and David Weenink)
  `http://www.fon.hum.uva.nl/praat/`
- evolOT: Simulating language evolution with OT (2005, Gerhard Jäger)
- etc.

Earlier ones: Optimality Interpreter (Apollo Hogan, 1993), OT Simple (Markus Walther, 1996), SA-OT demo (Biró 2005),. . . and many more.

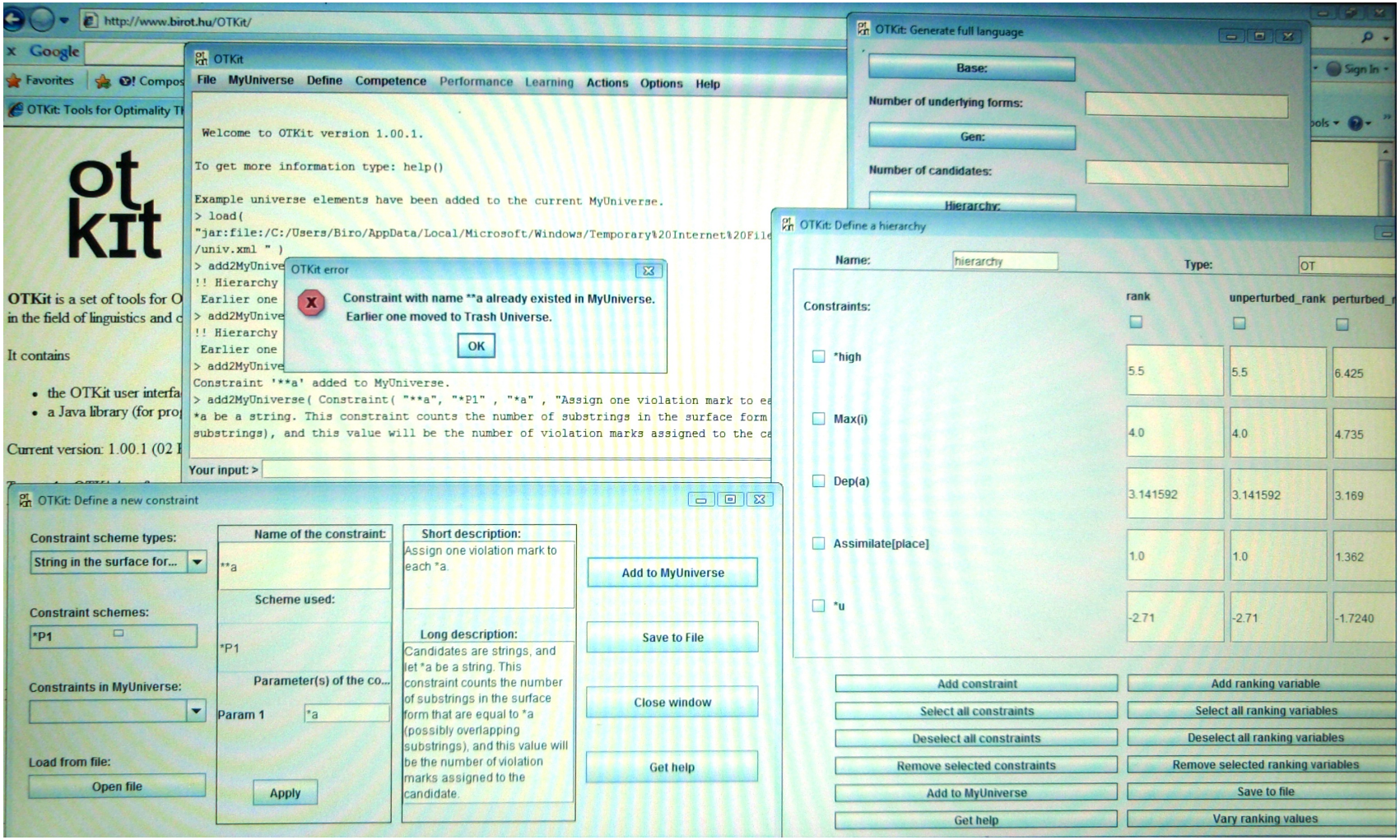
OTKit: Tools for Optimality Theory (Biró, 2010)

- Developed originally to teach myself Java. . .
- . . . and to support my own research.
- Also intended for colleagues and students.
- Hoping to develop once a course based on OTKit.
- Feedback appreciated!

OTKit: Tools for Optimality Theory (Biró, 2010)

- Java-based, platform-independent (Windows, Unix/Linux, Mac...).
- Graphical user interface for beginners.
- Scripting language and XML data structure for intermediate users.
- Java library for programmers: classes for forms, candidates, violations, constraints, hierarchies, Gen, production and learning algorithms, etc.
- Documentation: online help, manual, Javadoc, DTD.
- Available at `http://www.birot.hu/OTKit/`.

OTKit: Tools for Optimality Theory (Biró, 2010)



OTKit: Tools for Optimality Theory (Biró, 2010)

- Need of *explicitness*:
  what are the candidates? how many violations?
- Constraints as functions, not as desiderata.
  No need to list violations per candidates explicitly.
  User interface offers a large range of constraints.
- Possibly infinite candidate set.
  Find best candidate using simulated annealing.
  (Currently in Java library, yet to come in user interface.)
- Opportunity to generalize the notions ‘violation’, ‘candidate’, ‘ranking variable’, ‘learning step’.
  As we shall see: constraint arithmetic, ranking variable operations,. . .

Overview

Basic building blocks of OT

- Forms: e.g., elements of a set $\mathcal{F}$, and $\mathcal{F} := \mathcal{F}_u \cup \mathcal{F}_s$.
- Candidates: e.g., elements of $\mathcal{X} = \mathcal{F}_u \times \mathcal{F}_s$.
- Gen: a one-to-many mapping $\mathcal{F}_u \rightarrow \mathcal{X}$.

- Violations: some set $\mathcal{V}$, e.g. $\mathbb{N}_0$.
- Constraints: functions $\mathcal{X} \rightarrow \mathcal{V}$.

- Ranking values: some set $\mathcal{R}$, e.g., $\mathbb{R}$.
- Hierarchies: functions $\{C_i | i \in \mathcal{I}\} \rightarrow \mathcal{R}$.
- And many more: production methods, learning algorithms,...

Forms

- Atomic data structures.
- Can be used as underlying forms, surface forms,. . .
  and many more.
- Currently implemented: strings, counters, counters with strings.
- Possibilities in the future: metrical phonology trees,
  syntax trees, AVMs, etc.

Candidates

- Candidates $\neq$ surface forms,
  even if most often the distinction is ignored.
- Constraints $C(x)$ defined on candidates, and not on surface forms.
- Candidate is (underlying form, surface form) pair – most typically.
- Correspondence Theory (McCarthy and Prince 1995):
  (uf, sf, $\mathcal{R}$) triple, where $\mathcal{R}$ is a *correspondence relation*.
- Currently implemented:
  sf-only, (uf, sf) pair, multiple layers, chain of surface forms.

Gen

- Finite Gen from a predefined table.
- Given alphabet Σ, any underlying form mapped onto Σ^n or Σ^*.
- Predefined grammars:
  toy “string grammar”, metrical phonology.
- Gen arithmetic: the composition of two, predefined functions $\mathrm{Gen} = \mathrm{Gen}_1 \circ \mathrm{Gen}_2$.
- Do you have further suggestions?

Constraints

- Explicitly defined constraints:
  this candidate / surf. form / string is assigned so many violations.
- Define a constraint with a table.
- Penalize a specific substring:
  once or multiple times, if multiple occurrences in the sf string.
- Penalize substring on the left/right edges only.
- Alignment constraints (such as those in metrical phonology).
- Constraints on counters: return the value of that counter.
- Faithfulness (MAX, DEP) between uf and sf.
- Metrical phonology constraints.

Constraint arithmetics

- Constant function: $C(x) := c$ for all $x \in \mathrm{Gen}(u)$.
- Sum, product and ratio of two constraints: $C(x) := C_1(x) + C_2(x)$, $C(x) := C_1(x) \cdot C_2(x)$, $C(x) := C_1(x)/C_2(x)$.
- Maximum and minimum (disjunction and conjunction): $C(x) := \max(C_1(x), C_2(x))$, $C(x) := \min(C_1(x), C_2(x))$.
- Conditional constraints: $C(x) := \begin{cases} C_2(x) & \text{if } C_1(x) < 0, \\ C_3(x) & \text{if } C_1(x) = 0, \\ C_4(x) & \text{if } C_1(x) > 0. \end{cases}$
- Constraint applied to a modified surface string: temporarily remove or replace some substrings: $C(x) := C_1(\Phi(x))$.
  (E.g., temporarily remove Cs from word for a V harmony constraint.)

Hierarchies

- Constraints with ranking variables.
- Several ranking variables: rank, weight, perturbed rank, etc.
- Various production algorithms:
  standard OT, stochastic OT, HG, etc.
- Functions: generate tableaux (e.g., in LaTeXformat),
  factorial typology, etc.

Overview

What have we learned from implementing OT?

- OT as a linguistic model ?
- OT as a mathematical object !
- Forced to define everything explicitly.
- Forced to build up the building blocks from simple units. No *ad hoc* constraints.
- Overcoming the constant feeling of insecurity: *'Have I not made a mistake in my analysis? Are all candidates included? Have I correctly listed the violation marks in the tableau?'*

Thank you for your attention!

Tamás Biró:
tamas[dot]biro[at]btk[dot]elte[dot]hu
http://birot.web.elte.hu/, http://birot.hu/OTKit/

This research was supported by a *Veni Grant* (project number 275-89-004) offered by the Netherlands Organisation for Scientific Research (NWO), as well as by a *Marie Curie FP7 Integration Grant* (no. PCIG14-GA-2013-631599, "MeMoLI"), *7th EU Framework Programme*.