Heuristic Production, Heuristic Learning

Boltzmann distribution for strict domination

Tamás Biró

Yale University

NECPhon 7 @ MIT, October, 2013

Optimality Theory at a disciplinary crossroads

Theoretical linguistics → constraints

Computer science → optimization

Cognitive science

How to combine constraints into a single target function?

- Constraints: originally boolean functions (Chomsky).
  Most often, taking non-negative integer values. (*But P&S's* HNUC *takes sonority values!)*
- John von Neumann:
  "The brain does not use the language of mathematics."
- Convenience combined with correct predictions.

→ constraint combination: **strict domination** postulated.

Optimality Theory at a disciplinary crossroads

Theoretical linguistics → constraints

Computer science → optimization

Cognitive science

How to combine constraints into a single target function?

- Constraints: originally boolean functions (Chomsky).
  Most often, taking non-negative integer values. (*But P&S's* HNUC *takes sonority values!)*
- John von Neumann:
  "The brain does not use the language of mathematics."
- Convenience combined with correct predictions.

→ constraint combination: **strict domination** postulated.

Optimality Theory at a disciplinary crossroads

Theoretical linguistics → constraints

Computer science

→ optimization

Cognitive science

How to combine constraints into a single target function?

- Constraints: originally boolean functions (Chomsky).
  Most often, taking non-negative integer values. (*But P&S's* HNUC *takes sonority values!)*
- John von Neumann:
  "The brain does not use the language of mathematics."
- Convenience combined with correct predictions.

→ constraint combination: **strict domination** postulated.

Boltzmann distribution for optimization

- **Boltzmann distribution** (Gibbs distribution) in statistical physics:

$$\text{Prob}(\text{particle in state possessing energy } E_i) \propto e^{\frac{-E_i}{k_B T}}$$

- **Simulated annealing:** a general idea in AI
  - *Boltzmann machines*: a type of stochastic recurrent neural network (cf. Paul Smolensky's work).
  - A heuristic optimization technique for NP-hard problems (e.g., Černy 1985: traveling salesman problem).
  - A probability distribution family parameterized with T: $P(A) = \exp(-E(A)/T)/Z(T)$, where *partition function* $Z(T) = \sum_{A'} exp(-E(A')/T)$.

Boltzmann distribution for optimization

- **Boltzmann distribution** (Gibbs distribution) in statistical physics:

$$\text{Prob}(\text{particle in state possessing energy } E_i) \propto e^{\frac{-E_i}{k_B T}}$$

- **Simulated annealing:** a general idea in AI
  - *Boltzmann machines*: a type of stochastic recurrent neural network (cf. Paul Smolensky's work).
  - A heuristic optimization technique for NP-hard problems (e.g., Černy 1985: traveling salesman problem).
  - A probability distribution family parameterized with T: $P(A) = \exp(-E(A)/T)/Z(T)$, where *partition function* $Z(T) = \sum_{A'} exp(-E(A')/T)$.

Overview

1. Boltzmann distribution for OT: Performance
2. Boltzmann distribution for OT: Learning
3. How to define Boltzmann distribution for OT?
4. Conclusion

Overview

Stress patterns in (Dutch) fast speech

Forms and frequencies observed by Schreuder and Gilbers:

fo.to.toe.stel ‘camera’	*uit.ge.ve.rij* ‘publisher’	*stu.die.toe.la.ge* ‘study grant’	*per.fec.tio.nist* ‘perfectionist’
susu	ssus	susuu	usus
fó.to.tòe.stel slow: fast:	*ùit.gè.ve.ríj* slow: **0.96** fast: **0.67**	*stú.die.tòe.la.ge* slow: **0.81** fast: **0.38**	*per.fèc.tio.níst* slow: **0.20** fast: **0.13**
fó.to.toe.stèl slow: fast:	*ùit.ge.ve.ríj* slow: **0.04** fast: **0.33**	*stú.die.toe.là.ge* slow: **0.19** fast: **0.62**	*pèr.fec.tio.níst* slow: **0.80** fast: **0.87**
shift to right	beat reduction	shift to right	shift to left

Stress patterns in (Dutch) fast speech

Proposal of Schreuder and Gilbers:

Andante speech:

$u =$ /*foto+toestel*/		OUTPUT-OUTPUT CORRESPONDENCE	FOOT REPULSION	PARSE SYLLABLE
☞	(fóto)(tòestel)	0	1	0
	(fóto)toe(stèl)	1!	0	1

$\rightarrow$ Form *fótotòestel* in slow speech.

NB: Further output forms in the language: *fóto* and *tóestel*.

Stress patterns in (Dutch) fast speech

Proposal of Schreuder and Gilbers:

Allegro speech:

$u =$ /*foto+toestel*/	FOOT REPULSION	OUTPUT-OUTPUT CORRESPONDENCE	PARSE SYLLABLE
(fóto)(tòestel)	1!	0	0
☞ (fóto)toe(stèl)	0	1	1

→ Form *fótotoestèl* in fast speech.

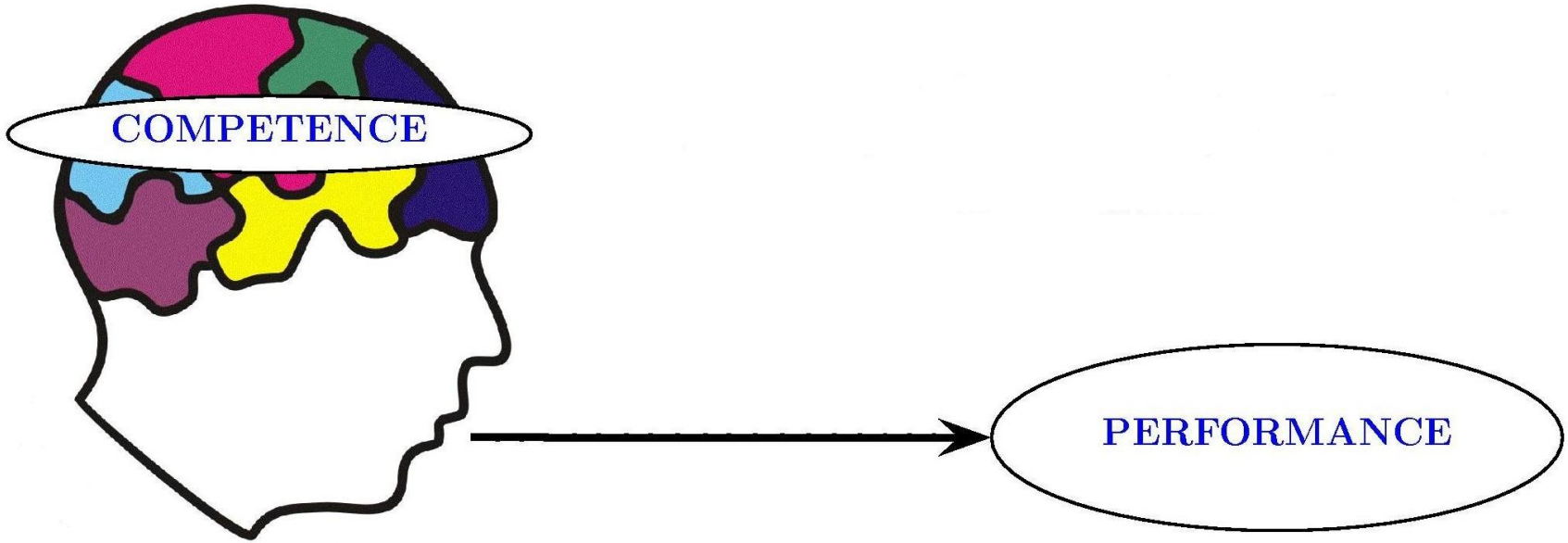
NB: Further output forms in the language: *fóto* and *tóestel*.

Stress patterns in (Dutch) fast speech

- Different grammar for fast speech?
  - Interpretation: different competence during fast speech.
  - Prediction: abrupt switch, 0% vs. 100% pattern.

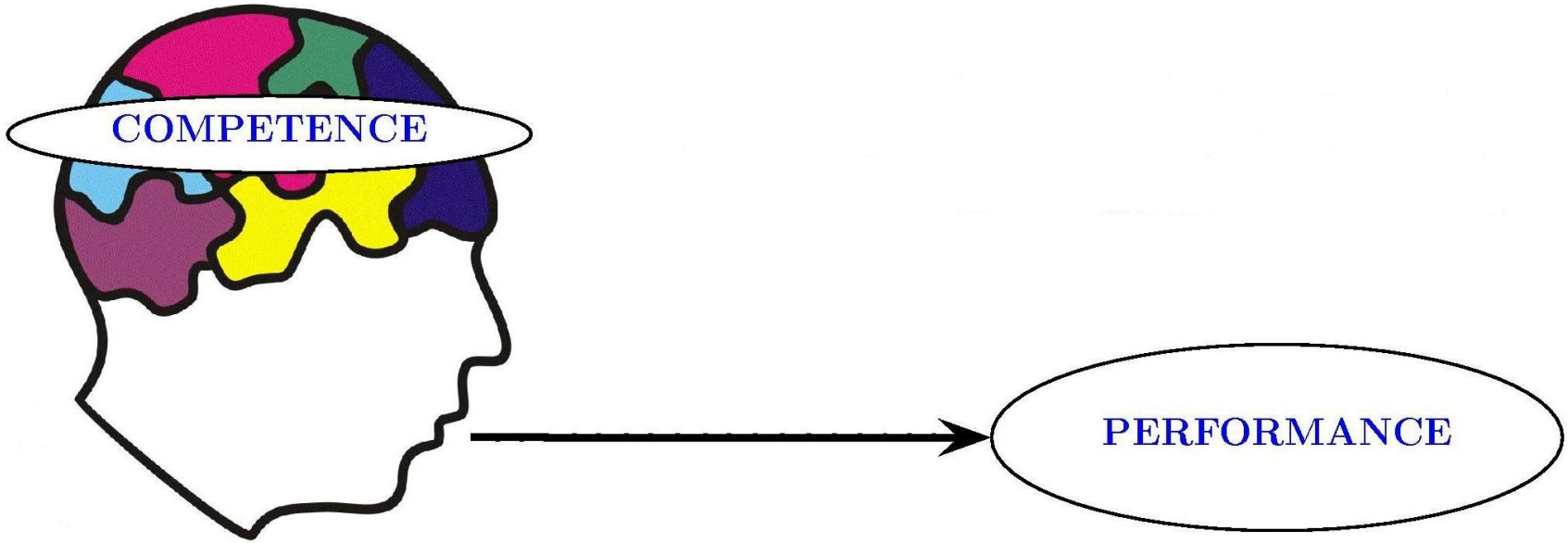
- Boersma's Stochastic OT
  - Assumption: larger noise in fast speech, causing more frequent re-ranking.
  - Prediction: same distributional pattern for all word types.
  - Prediction: never more than 50% fast speech form rate.

Errors in mental computation



static knowledge — processing — in the brain
Optimality Theory — Simulated Annealing for OT

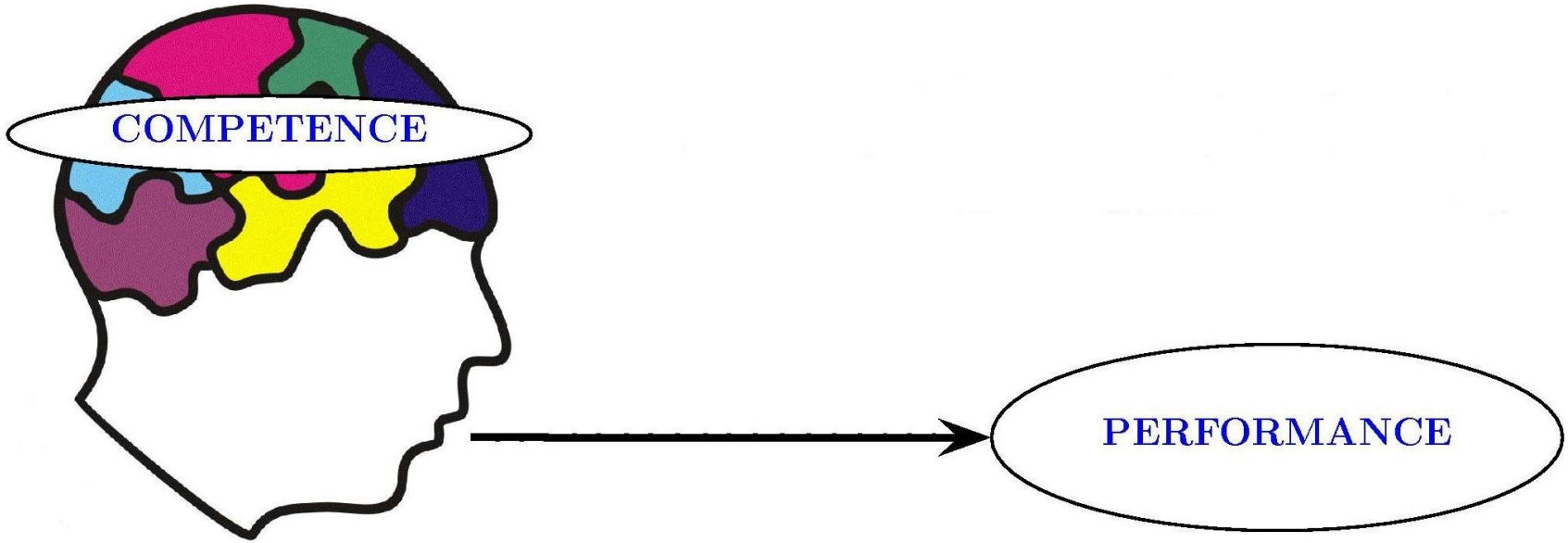
Errors in mental computation



static knowledge — processing — in the brain

Optimality Theory — Simulated Annealing for OT

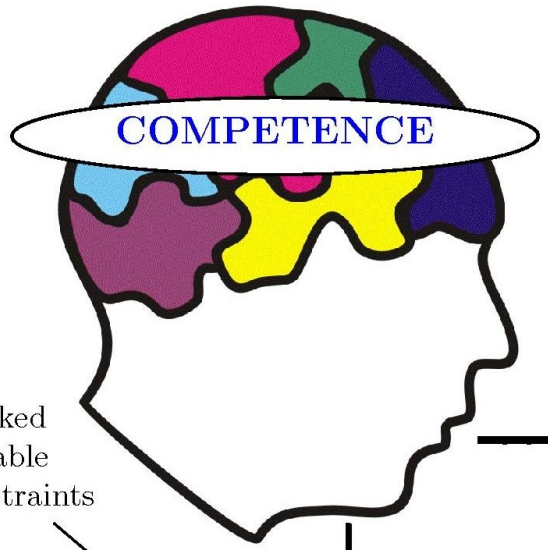
Errors in mental computation



static knowledge — processing — in the brain

Optimality Theory — Simulated Annealing for OT

Errors in computation: erroneous implementation



A grammar is a Harmony function on the candidate set, defined by the ranked constraints.
Global optimum: more harmonic than all other candidates.
Local optimum: more harmonic than its neighbours.

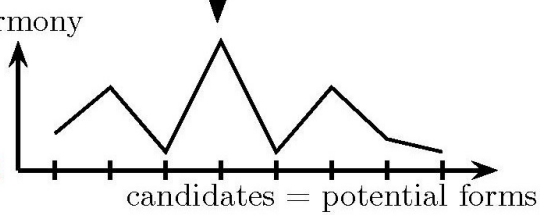
Ranked violable constraints

Candidates (potential forms)

PERFORMANCE

/aat/	NoCoda	Parse	Onset
☞ [a.a<t>]		*	**
[a.at]	*!		**
[<a>at]	*!	*	*

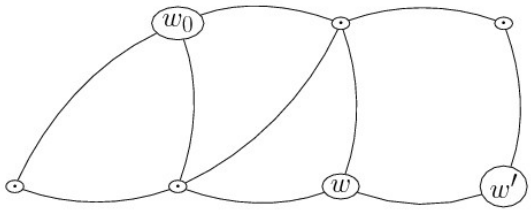
implementation

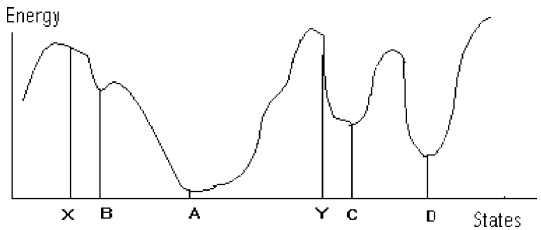


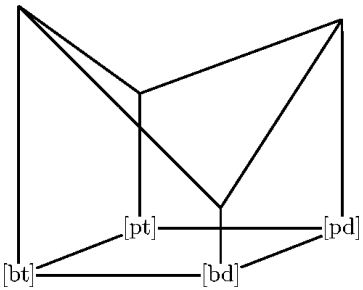
Optimality Theory
grammar
competence model
grammatical form = ☞ (globally) optimal candidate

SA-OT
implementation
performance model
produced forms = globally or locally optimal candidates

Basic idea of Simulated Annealing



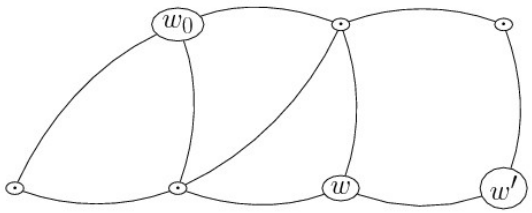


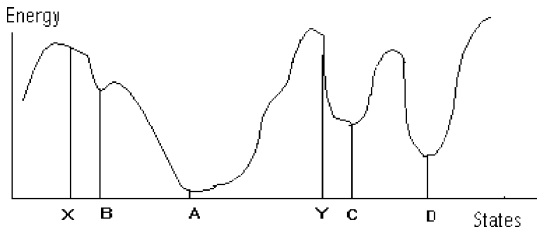


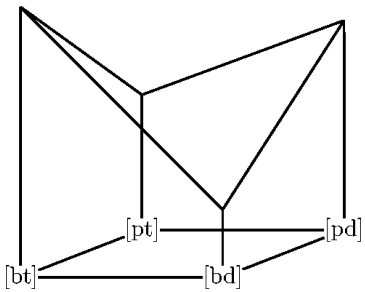
Step 1 – introducing landscape:

- Horizontal: universal *neighborhood structure* (a.k.a. *topology*) on the universal candidate set $\mathrm{Gen}(u)$.
- Vertical: grammar-dependent harmony H.
- Random walk in this landscape.

Basic idea of Simulated Annealing







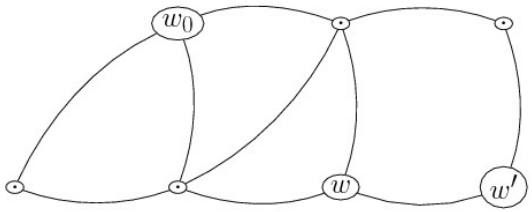
Step 2 – walking in this landscape:

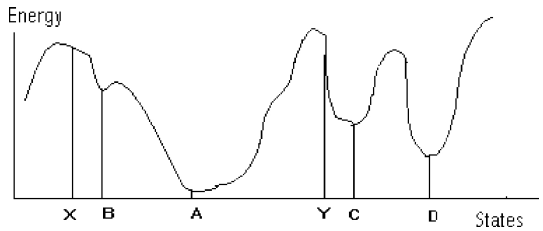
- Pick a random neighbor of your position.
- If neighbor is more optimal: move.
- If less optimal: move in the beginning, don't move later.
- Exponential transition probability

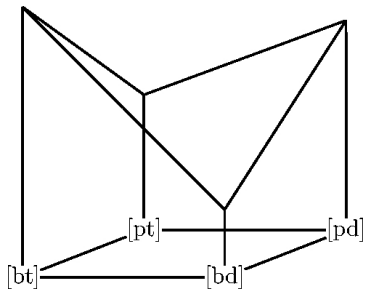
$$P(f \to f' | T) = e^{-\frac{H(f') - H(f)}{T}}$$

Q: How to define it, given the non-numeric target function?

Basic idea of Simulated Annealing







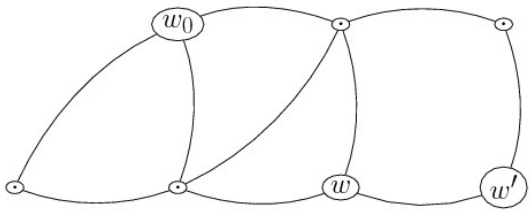
Step 2 – walking in this landscape:

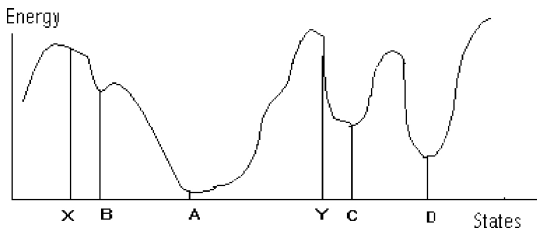
- Pick a random neighbor of your position.
- If neighbor is more optimal: move.
- If less optimal: move in the beginning, don't move later.
- Exponential transition probability

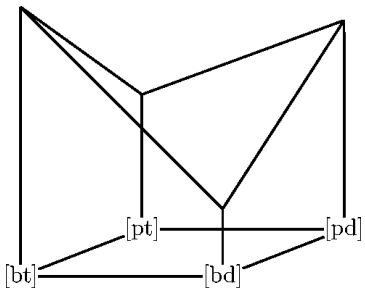
$$P(f \to f' | T) = e^{-\frac{H(f')-H(f)}{T}}$$

Q: How to define it, given the non-numeric target function?

Basic idea of Simulated Annealing



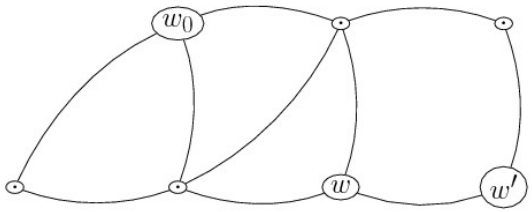


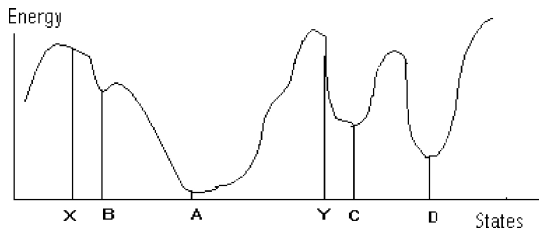


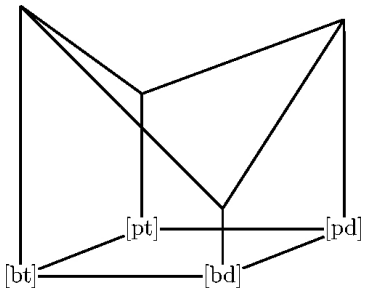
Step 3 – performing a random walk on this landscape:

- Start random walk from some initial position.
- End position returned as output of the algorithm: produced form.
- Hopefully, global optimum (grammatical form) is found. Yet...
- Neighborhood structure $\rightarrow$ local optima,
  where random walker can get stuck. Performance errors.

Basic idea of Simulated Annealing





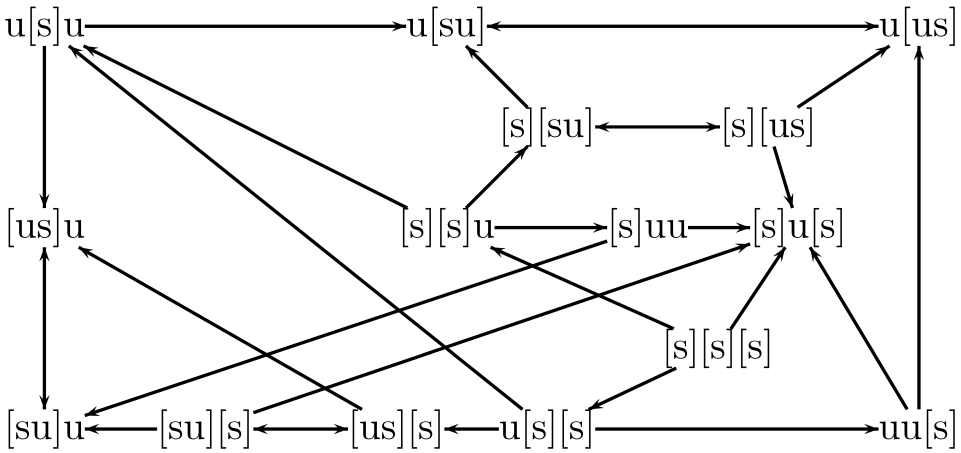


Step 4 – Precision of the algorithm

- **Precision** of the algorithm: chance of ending up in global optimum, and hence returning grammatical form.
- Precision of the algorithm depends on its speed.
- Trade precision for speed – just like human mind!
- Due to strict domination: slow annealing not necessarily guarantees 100% precision! (irregular forms)

How to predict stress pattern in (Dutch) fast speech?

Landscape:

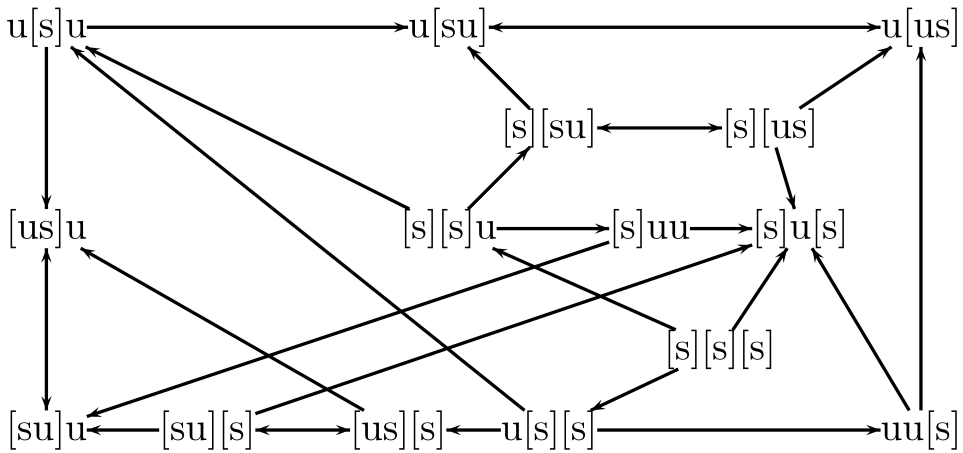


Basic steps that connect neighbors:

- Move foot boundary: [s] $\Leftrightarrow$ [su]; [s] $\Leftrightarrow$ [us].
- Change head of foot: [su] $\Leftrightarrow$ [us].
- Insert/delete monosyllabic foot: [s] $\Leftrightarrow$ u

How to predict stress pattern in (Dutch) fast speech?

Hierarchy:



ALIGN-LEFT ≫ $OOC_{z=2}$ ≫ FOOTREPULSION ≫ PARSESYLL ≫ TROCHAIC

- Global optimum: [s]u[su].
- Local optima: [s]u[su] and [su]u[s].
- Local optimum [su]u[s] has less harmonic neighbors: [su]uu, [su][s][s], [us]u[s], [s]uu[s], [su][us].

How to predict stress pattern in (Dutch) fast speech?

Local optimum [su]u[s] has less harmonic neighbors:
[su]uu, [su][s][s], [us]u[s], [s]uu[s], [su][us].

/fototoestel/	ALIGN-LEFT	$OOC_{z=2}$	FTREPULS	PARSES	TROCHAIC
∼ [su]u[s]	0	2	0	1	0
[su]uu	0	3	0	2	0
[su][s][s]	0	3	2	0	0
[us]u[s]	0	4	0	1	1
[s]uu[s]	0	2	0	2	0
[su][us]	0	2	1	0	1

How to predict stress pattern in (Dutch) fast speech?

Simulated Annealing for Optimality Theory: demo page

`http://www.birot.hu/sa-ot/`

OTKit Tools for Optimality Theory

`http://www.birot.hu/OTKit/`

How to predict stress pattern in (Dutch) fast speech?

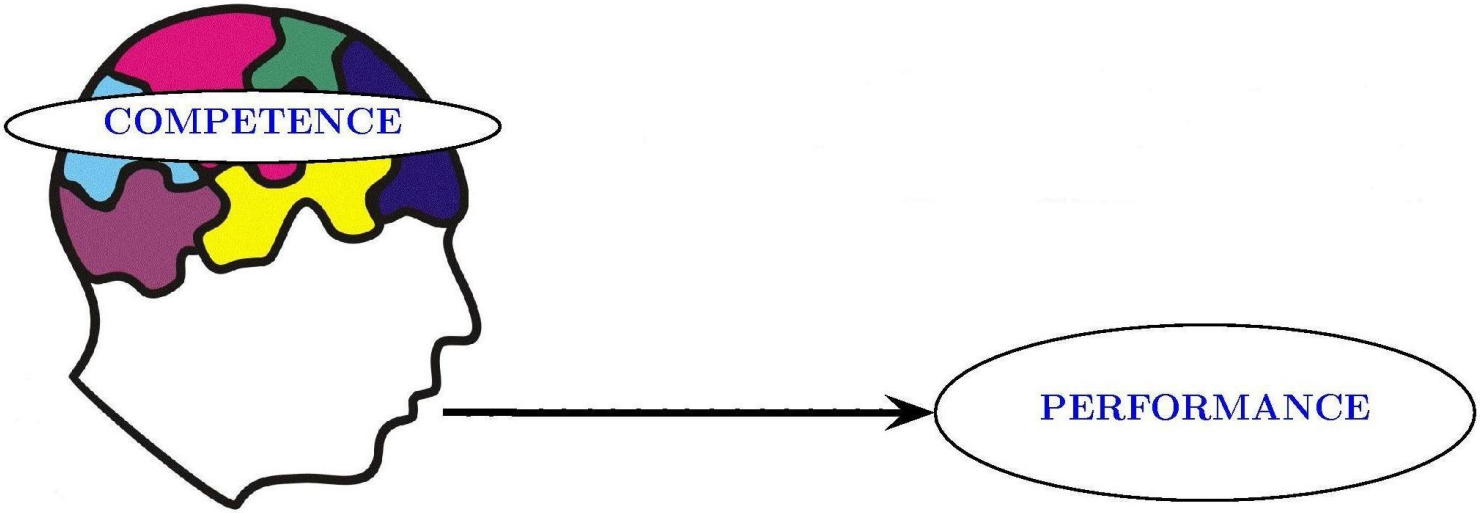
fo.to.toe.stel ‘camera’	*uit.ge.ve.rij* ‘publisher’	*stu.die.toe.la.ge* ‘study grant’	*per.fec.tio.nist* ‘perfectionist’
susu	ssus	susuu	usus
fó.to.tòe.stel slow: *1.00* fast: *0.82*	*ùit.gè.ve.ríj* slow: *0.97* / **0.96** fast: *0.65* / **0.67**	*stú.die.tòe.la.ge* slow: *0.96* / **0.81** fast: *0.55* / **0.38**	*per.fèc.tio.níst* slow: *0.91* / **0.20** fast: *0.49* / **0.13**
fó.to.toe.stèl slow: *0.00* fast: *0.18*	*ùit.ge.ve.ríj* slow: *0.03* / **0.04** fast: *0.35* / **0.33**	*stú.die.toe.là.ge* slow: *0.04* / **0.19** fast: *0.45* / **0.62**	*pèr.fec.tio.níst* slow: *0.07* / **0.80** fast: *0.39* / **0.87**

Simulated / **observed** (Schreuder) frequencies.

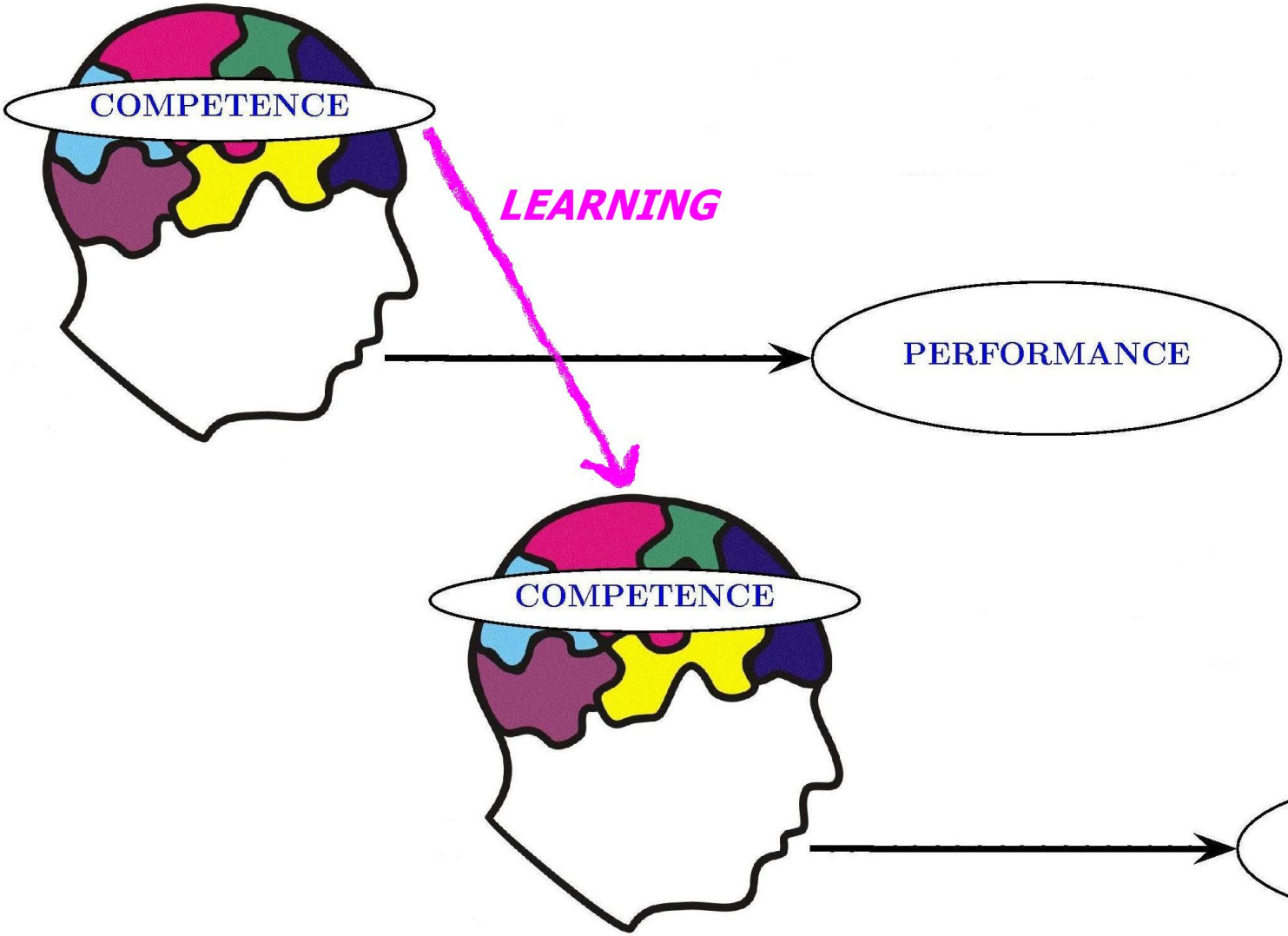
In the simulations, $T_{step} = 3$ used for fast speech and $T_{step} = 0.1$ for slow speech.

Overview

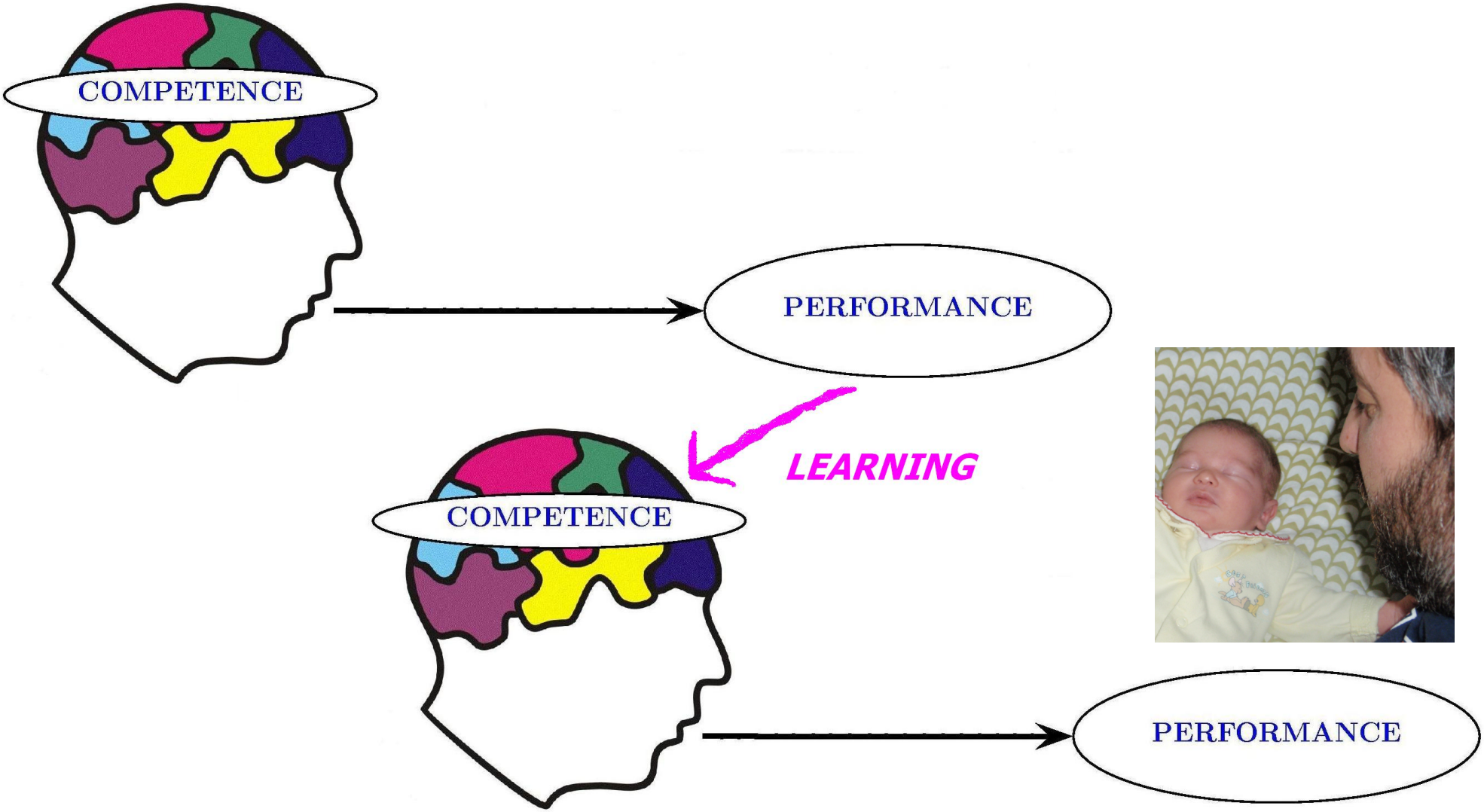
The language acquisition problem



Learning from competence?



Learning from performance!



Error-Driven Constraint Demotion (Tesar and Smolensky)

		/ab.ra.ka.dab.ra/	NONFINAL	TROCHAIC	FOOTREPULSION
l	1.	*ab.ra.[ka.dáb].ra*	0	1	0
	2.	*[àb.ra].[ka.dáb].ra*	0	1	1
w	3.	*[àb.ra].ka.[dáb.ra]*	1	0	0

- Teacher: FOOTREPULSION ≫ TROCHAIC ≫ NONFINAL, producing grammatical form w: *[àb.ra].ka.[dáb.ra]*.
- Learner: NONFINAL ≫ TROCHAIC ≫ FOOTREPULSION, producing loser: *ab.ra.[ka.dáb].ra*.
- Compare w and l:
- Promote w-preferring constraints: TROCHAIC.
  Demote l-preferring constraints: NONFINAL.
- Learner's new grammar will produce form w: *[àb.ra].ka.[dáb.ra]*: with grammar TROCHAIC ≫ NONFINAL ≫ FOOTREPULSION.

Error-Driven Constraint Demotion (Tesar and Smolensky)

		/ab.ra.ka.dab.ra/	NONFINAL	TROCHAIC	FOOTREPULSION
l	1.	*ab.ra.[ka.dáb].ra*	0	1	0
	2.	*[àb.ra].[ka.dáb].ra*	0	1	1
w	3.	*[àb.ra].ka.[dáb.ra]*	1	0	0

- Teacher: FOOTREPULSION ≫ TROCHAIC ≫ NONFINAL, producing grammatical form w: *[àb.ra].ka.[dáb.ra]*.
- Learner: NONFINAL ≫ TROCHAIC ≫ FOOTREPULSION, producing loser: *ab.ra.[ka.dáb].ra*.
- Compare w and l:
- Promote w-preferring constraints: TROCHAIC.
  Demote l-preferring constraints: NONFINAL.
- Learner's new grammar will produce form w: *[àb.ra].ka.[dáb.ra]*: with grammar TROCHAIC ≫ NONFINAL ≫ FOOTREPULSION.

Error-Driven Constraint Demotion (Tesar and Smolensky)

		/ab.ra.ka.dab.ra/	NONFINAL	TROCHAIC	FOOTREPULSION
l	1.	*ab.ra.[ka.dáb].ra*	0	1	0
	2.	*[àb.ra].[ka.dáb].ra*	0	1	1
w	3.	*[àb.ra].ka.[dáb.ra]*	1	0	0

- Teacher: FOOTREPULSION ≫ TROCHAIC ≫ NONFINAL, producing grammatical form w: *[àb.ra].ka.[dáb.ra]*.
- Learner: NONFINAL ≫ TROCHAIC ≫ FOOTREPULSION, producing loser: *ab.ra.[ka.dáb].ra*.
- Compare w and l:
- Promote w-preferring constraints: TROCHAIC.
  Demote l-preferring constraints: NONFINAL.
- Learner's new grammar will produce form w: *[àb.ra].ka.[dáb.ra]*: with grammar TROCHAIC ≫ NONFINAL ≫ FOOTREPULSION.

Error-Driven Constraint Demotion (Tesar and Smolensky)

		/ab.ra.ka.dab.ra/	NONFINAL	TROCHAIC	FOOTREPULSION
l	1.	*ab.ra.[ka.dáb].ra*	0	1	0
	2.	*[àb.ra].[ka.dáb].ra*	0	1	1
w	3.	*[àb.ra].ka.[dáb.ra]*	1	0	0

- Teacher: FOOTREPULSION ≫ TROCHAIC ≫ NONFINAL, producing grammatical form w: *[àb.ra].ka.[dáb.ra]*.
- Learner: NONFINAL ≫ TROCHAIC ≫ FOOTREPULSION, producing loser: *ab.ra.[ka.dáb].ra*.
- Compare w and l:
- Promote w-preferring constraints: TROCHAIC.
  Demote l-preferring constraints: NONFINAL.
- Learner's new grammar will produce form w: *[àb.ra].ka.[dáb.ra]*: with grammar TROCHAIC ≫ NONFINAL ≫ FOOTREPULSION.

Robust Interpretive Parsing (Tesar and Smolensky)

		/ab.ra.ka.dab.ra/	NONFINAL	TROCHAIC	FOOTREPULSION
l	1.	*ab.ra.[ka.dáb].ra*	0	1	0
w	2.	*[àb.ra].[ka.dáb].ra*	0	1	1
☞	3.	*[àb.ra].ka.[dáb.ra]*	1	0	0

- Teacher: FOOTREPULSION ≫ TROCHAIC ≫ NONFINAL, producing grammatical form ☞: *[àb.ra].ka.[dáb.ra]*.
- Learner: NONFINAL ≫ TROCHAIC ≫ FOOTREPULSION, producing loser: *ab.ra.[ka.dáb].ra*.
- Learner hears *àb.ra.ka.dáb.ra*! Two possible candidates. The winner must have been, the best one, *[àb.ra].[ka.dáb].ra*.
- Compare w and l. Promote w-preferring constraints: none. Demote l-preferring constraints: FOOTREPULSION → deadlock!

Robust Interpretive Parsing (Tesar and Smolensky)

		/ab.ra.ka.dab.ra/	NONFINAL	TROCHAIC	FOOTREPULSION
l	1.	*ab.ra.[ka.dáb].ra*	0	1	0
w	2.	*[àb.ra].[ka.dáb].ra*	0	1	1
☞	3.	*[àb.ra].ka.[dáb.ra]*	1	0	0

- Teacher: FOOTREPULSION ≫ TROCHAIC ≫ NONFINAL, producing grammatical form ☞: *[àb.ra].ka.[dáb.ra]*.
- Learner: NONFINAL ≫ TROCHAIC ≫ FOOTREPULSION, producing loser: *ab.ra.[ka.dáb].ra*.
- Learner hears *àb.ra.ka.dáb.ra*! Two possible candidates. The winner must have been, the best one, *[àb.ra].[ka.dáb].ra*.
- Compare w and l. Promote w-preferring constraints: none. Demote l-preferring constraints: FOOTREPULSION → deadlock!

Robust Interpretive Parsing (Tesar and Smolensky)

		/ab.ra.ka.dab.ra/	NONFINAL	TROCHAIC	FOOTREPULSION
l	1.	*ab.ra.[ka.dáb].ra*	0	1	0
w	2.	*[àb.ra].[ka.dáb].ra*	0	1	1
☞	3.	*[àb.ra].ka.[dáb.ra]*	1	0	0

- Teacher: FOOTREPULSION ≫ TROCHAIC ≫ NONFINAL, producing grammatical form ☞: *[àb.ra].ka.[dáb.ra]*.
- Learner: NONFINAL ≫ TROCHAIC ≫ FOOTREPULSION, producing loser: *ab.ra.[ka.dáb].ra*.
- Learner hears *àb.ra.ka.dáb.ra*! Two possible candidates. The winner must have been, the best one, *[àb.ra].[ka.dáb].ra*.
- Compare w and l. Promote w-preferring constraints: none. Demote l-preferring constraints: FOOTREPULSION → deadlock!

Robust Interpretive Parsing (Tesar and Smolensky)

		/ab.ra.ka.dab.ra/	NONFINAL	TROCHAIC	FOOTREPULSION
l	1.	*ab.ra.[ka.dáb].ra*	0	1	0
w	2.	*[àb.ra].[ka.dáb].ra*	0	1	1
☞	3.	*[àb.ra].ka.[dáb.ra]*	1	0	0

- Teacher: FOOTREPULSION ≫ TROCHAIC ≫ NONFINAL, producing grammatical form ☞: *[àb.ra].ka.[dáb.ra]*.
- Learner: NONFINAL ≫ TROCHAIC ≫ FOOTREPULSION, producing loser: *ab.ra.[ka.dáb].ra*.
- Learner hears *àb.ra.ka.dáb.ra*! Two possible candidates. The winner must have been, the best one, *[àb.ra].[ka.dáb].ra*.
- Compare w and l. Promote w-preferring constraints: none. Demote l-preferring constraints: FOOTREPULSION → deadlock!

Robust Interpretive Parsing (Tesar and Smolensky)

		/ab.ra.ka.dab.ra/	NONFINAL	TROCHAIC	FOOTREPULSION
l	1.	*ab.ra.[ka.dáb].ra*	0	1	0
w	2.	*[àb.ra].[ka.dáb].ra*	0	1	1
☞	3.	*[àb.ra].ka.[dáb.ra]*	1	0	0

- Teacher: FOOTREPULSION ≫ TROCHAIC ≫ NONFINAL, producing grammatical form ☞: *[àb.ra].ka.[dáb.ra]*.
- Learner: NONFINAL ≫ TROCHAIC ≫ FOOTREPULSION, producing loser: *ab.ra.[ka.dáb].ra*.
- Learner hears *àb.ra.ka.dáb.ra*! Two possible candidates. The winner must have been, the best one, *[àb.ra].[ka.dáb].ra*.
- Compare w and l. Promote w-preferring constraints: none. Demote l-preferring constraints: FOOTREPULSION → deadlock!

Revised *Robust Interpretive Parsing* (Biró, 2013)

		/ab.ra.ka.dab.ra/	NONFINAL	TROCHAIC	FOOTREPULSION
l	1.	*ab.ra.[ka.dáb].ra*	0	1	0
	2.	*[àb.ra].[ka.dáb].ra*	0	1	1
☞	3.	*[àb.ra].ka.[dáb.ra]*	1	0	0
w			0.5	0.5	0.5

- Teacher: FOOTREPULSION ≫ TROCHAIC ≫ NONFINAL, → ☞*[àb.ra].ka.[dáb.ra]*.
- Learner: NONFINAL ≫ TROCHAIC ≫ FOOTREPULSION, → l *ab.ra.[ka.dáb].ra*.
- Learner: w is either *[àb.ra].[ka.dáb].ra* or *[àb.ra].ka.[dáb.ra]*.
- Calculate (weighted) average, as *winner violation profile*. Compare it to loser. Promote w-preferring constraints: TROCHAIC. Demote l-preferring constraints: FOOTREPULSION and NONFINAL.
- → solution: TROCHAIC ≫ NONFINAL ≫ FOOTREPULSION. *Learner's new grammar different from, but equivalent to teacher's!*

Revised *Robust Interpretive Parsing* (Biró, 2013)

		/ab.ra.ka.dab.ra/	NONFINAL	TROCHAIC	FOOTREPULSION
l	1.	*ab.ra.[ka.dáb].ra*	0	1	0
	2.	*[àb.ra].[ka.dáb].ra*	0	1	1
☞	3.	*[àb.ra].ka.[dáb.ra]*	1	0	0
w			0.5	0.5	0.5

- Teacher: FOOTREPULSION ≫ TROCHAIC ≫ NONFINAL, → ☞*[àb.ra].ka.[dáb.ra]*.
- Learner: NONFINAL ≫ TROCHAIC ≫ FOOTREPULSION, → l *ab.ra.[ka.dáb].ra*.
- Learner: w is either *[àb.ra].[ka.dáb].ra* or *[àb.ra].ka.[dáb.ra]*.
- Calculate (weighted) average, as *winner violation profile*. Compare it to loser. Promote w-preferring constraints: TROCHAIC. Demote l-preferring constraints: FOOTREPULSION and NONFINAL.
- → solution: TROCHAIC ≫ NONFINAL ≫ FOOTREPULSION. *Learner's new grammar different from, but equivalent to teacher's!*

Revised *Robust Interpretive Parsing* (Biró, 2013)

		/ab.ra.ka.dab.ra/	NONFINAL	TROCHAIC	FOOTREPULSION
l	1.	*ab.ra.[ka.dáb].ra*	0	1	0
	2.	*[àb.ra].[ka.dáb].ra*	0	1	1
☞	3.	*[àb.ra].ka.[dáb.ra]*	1	0	0
w			0.5	0.5	0.5

- Teacher: FOOTREPULSION ≫ TROCHAIC ≫ NONFINAL, → ☞*[àb.ra].ka.[dáb.ra]*.
- Learner: NONFINAL ≫ TROCHAIC ≫ FOOTREPULSION, → l *ab.ra.[ka.dáb].ra*.
- Learner: w is either *[àb.ra].[ka.dáb].ra* or *[àb.ra].ka.[dáb.ra]*.
- Calculate (weighted) average, as *winner violation profile*.
  Compare it to loser. Promote w-preferring constraints: TROCHAIC.
  Demote l-preferring constraints: FOOTREPULSION and NONFINAL.
- → solution: TROCHAIC ≫ NONFINAL ≫ FOOTREPULSION.
  Learner's new grammar different from, but equivalent to teacher's!

Revised *Robust Interpretive Parsing* (Biró, 2013)

		/ab.ra.ka.dab.ra/	NONFINAL	TROCHAIC	FOOTREPULSION
l	1.	*ab.ra.[ka.dáb].ra*	0	1	0
	2.	*[àb.ra].[ka.dáb].ra*	0	1	1
☞	3.	*[àb.ra].ka.[dáb.ra]*	1	0	0
w			0.5	0.5	0.5

- Teacher: FOOTREPULSION ≫ TROCHAIC ≫ NONFINAL, → ☞*[àb.ra].ka.[dáb.ra]*.
- Learner: NONFINAL ≫ TROCHAIC ≫ FOOTREPULSION, → l *ab.ra.[ka.dáb].ra*.
- Learner: w is either *[àb.ra].[ka.dáb].ra* or *[àb.ra].ka.[dáb.ra]*.
- Calculate (weighted) average, as *winner violation profile*. Compare it to loser. Promote w-preferring constraints: TROCHAIC. Demote l-preferring constraints: FOOTREPULSION and NONFINAL.
- → solution: TROCHAIC ≫ NONFINAL ≫ FOOTREPULSION.
  Learner's new grammar different from, but equivalent to teacher's!

Revised *Robust Interpretive Parsing* (Biró, 2013)

		/ab.ra.ka.dab.ra/	NONFINAL	TROCHAIC	FOOTREPULSION
l	1.	*ab.ra.[ka.dáb].ra*	0	1	0
	2.	*[àb.ra].[ka.dáb].ra*	0	1	1
☞	3.	*[àb.ra].ka.[dáb.ra]*	1	0	0
w			0.5	0.5	0.5

- Teacher: FOOTREPULSION ≫ TROCHAIC ≫ NONFINAL, → ☞*[àb.ra].ka.[dáb.ra]*.
- Learner: NONFINAL ≫ TROCHAIC ≫ FOOTREPULSION, → l *ab.ra.[ka.dáb].ra*.
- Learner: w is either *[àb.ra].[ka.dáb].ra* or *[àb.ra].ka.[dáb.ra]*.
- Calculate (weighted) average, as *winner violation profile*. Compare it to loser. Promote w-preferring constraints: TROCHAIC. Demote l-preferring constraints: FOOTREPULSION and NONFINAL.
- → solution: TROCHAIC ≫ NONFINAL ≫ FOOTREPULSION.

Learner's new grammar different from, but equivalent to teacher's!

Revised *Robust Interpretive Parsing* (Biró, 2013)

		/ab.ra.ka.dab.ra/	NONFINAL	TROCHAIC	FOOTREPULSION
l	1.	*ab.ra.[ka.dáb].ra*	0	1	0
	2.	*[àb.ra].[ka.dáb].ra*	0	1	1
☞	3.	*[àb.ra].ka.[dáb.ra]*	1	0	0
w			0.5	0.5	0.5

- Teacher: FOOTREPULSION ≫ TROCHAIC ≫ NONFINAL, → ☞*[àb.ra].ka.[dáb.ra]*.
- Learner: NONFINAL ≫ TROCHAIC ≫ FOOTREPULSION, → l *ab.ra.[ka.dáb].ra*.
- Learner: w is either *[àb.ra].[ka.dáb].ra* or *[àb.ra].ka.[dáb.ra]*.
- Calculate (weighted) average, as *winner violation profile*. Compare it to loser. Promote w-preferring constraints: TROCHAIC. Demote l-preferring constraints: FOOTREPULSION and NONFINAL.
- → solution: TROCHAIC ≫ NONFINAL ≫ FOOTREPULSION.
  Learner's new grammar different from, but equivalent to teacher's!

Revised *Robust Interpretive Parsing* (Biró, 2013)

- $\mathrm{RipSet}(o)$: set of possible interpretations of over form o.
- **Weighted average violation** of constraint C:

$$C\left(\mathrm{RipSet}(o)\right) = \sum_{f \in \mathrm{RipSet}(o)} w(f) \cdot C(f)$$

- where weights originate from a Boltzmann distribution at temperature T:

$$w(f|T) = \frac{e^{-H(f)/T}}{Z(T)}, \qquad \text{that is} \qquad \frac{1}{w(f)} = \sum_{f' \in \mathrm{RipSet}(o)} e^{-\frac{H(f')-H(f)}{T}}$$

- "Very large" T: equal weight to all possible interpretations.
  "Very small" T: traditional RIP (only most harmonic counts).

- **Q: But how to calculate it, since $H(f)$ is a vector??**

Revised *Robust Interpretive Parsing* (Biró, 2013)

- $\mathrm{RipSet}(o)$: set of possible interpretations of over form o.
- **Weighted average violation** of constraint C:

$$C\left(\mathrm{RipSet}(o)\right) = \sum_{f \in \mathrm{RipSet}(o)} w(f) \cdot C(f)$$

- where weights originate from a Boltzmann distribution at temperature T:

$$w(f|T) = \frac{e^{-H(f)/T}}{Z(T)}, \qquad \text{that is} \qquad \frac{1}{w(f)} = \sum_{f' \in \mathrm{RipSet}(o)} e^{-\frac{H(f')-H(f)}{T}}$$

- "Very large" T: equal weight to all possible interpretations.
  "Very small" T: traditional RIP (only most harmonic counts).

- **Q: But how to calculate it, since $H(f)$ is a vector??**

Revised *Robust Interpretive Parsing* (Biró, 2013)

- $\mathrm{RipSet}(o)$: set of possible interpretations of over form o.
- **Weighted average violation** of constraint C:

$$C\left(\mathrm{RipSet}(o)\right) = \sum_{f \in \mathrm{RipSet}(o)} w(f) \cdot C(f)$$

- where weights originate from a Boltzmann distribution at temperature T:

$$w(f|T) = \frac{e^{-H(f)/T}}{Z(T)}, \qquad \text{that is} \qquad \frac{1}{w(f)} = \sum_{f' \in \mathrm{RipSet}(o)} e^{-\frac{H(f')-H(f)}{T}}$$

- "Very large" T: equal weight to all possible interpretations.
  "Very small" T: traditional RIP (only most harmonic counts).

- **Q: But how to calculate it, since $H(f)$ is a vector??**

Revised *Robust Interpretive Parsing* (Biró, 2013)

- $\mathrm{RipSet}(o)$: set of possible interpretations of over form o.
- **Weighted average violation** of constraint C:

$$C\left(\mathrm{RipSet}(o)\right) = \sum_{f \in \mathrm{RipSet}(o)} w(f) \cdot C(f)$$

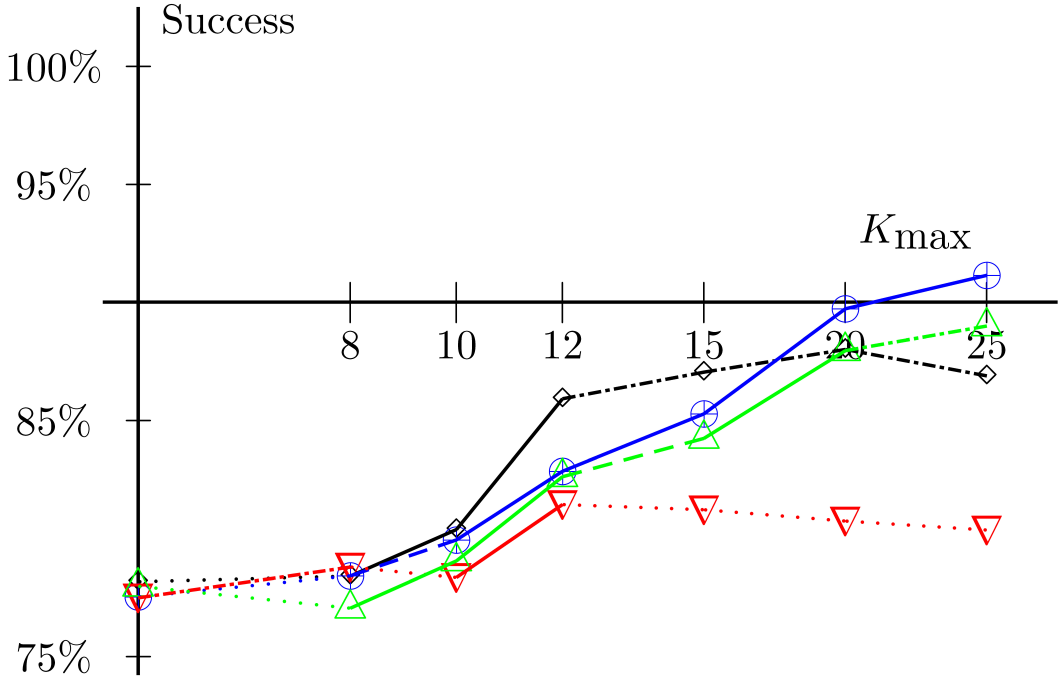
- where weights originate from a Boltzmann distribution at temperature T:

$$w(f|T) = \frac{e^{-H(f)/T}}{Z(T)}, \qquad \text{that is} \qquad \frac{1}{w(f)} = \sum_{f' \in \mathrm{RipSet}(o)} e^{-\frac{H(f')-H(f)}{T}}$$

- "Very large" T: equal weight to all possible interpretations.
  "Very small" T: traditional RIP (only most harmonic counts).

- **Q: But how to calculate it, since $H(f)$ is a vector??**

Revised *Robust Interpretive Parsing* (Biró, 2013)



Success rate of learning a random target grammar, as a function of parameter K_{max}, for different update rules. Random initial grammar and random target grammar, with twelve constraints.

Overview

Optimality Theory formalized

- Underlying form $u \mapsto$ a set of candidates $\mathrm{Gen}(u)$,
- Corresponding to underlying form u, grammar H predicts grammatical surface form SF:

$$\mathrm{SF}(u) = \underset{f \in \mathrm{Gen}(u)}{\arg\,\mathrm{opt}} \; H(f)$$

Optimality Theory formalized

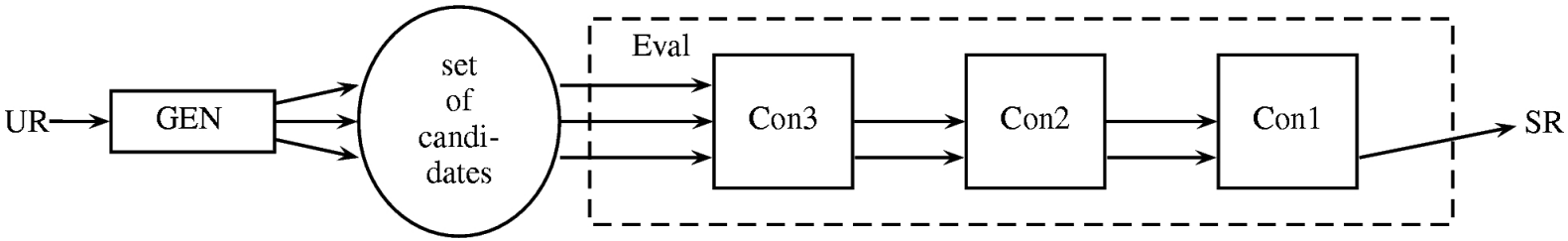
- Underlying form $u \mapsto$ a set of candidates $\mathrm{Gen}(u)$,

- Corresponding to underlying form u,
  grammar H predicts grammatical surface form SF:

$$\mathrm{SF}(u) = \underset{f \in \mathrm{Gen}(u)}{\arg\,\mathrm{opt}} \; H(f)$$

Optimality Theory formalized

- What is $H(f)$?



- Harmony function H to be optimized is non-numeric:

$$H(f) = \left(C_{n-1}(f), C_{n-2}(f), \ldots, C_0(f)\right)$$

- Optimization by **lexicographic order**!

Calculating exponentials?

Q: divide and exponentiate vectors, resulting in real numbers.

- Transition probabilities
  in the *Simulated Annealing for OT Algorithm*:

$$P(f \rightarrow f' \,|\, T) = e^{-\frac{H(f')-H(f)}{T}}$$

- Boltzmann weights
  in the *Generalized Robust Interpretive Parsing Algorithm*:

$$\frac{1}{w(f)} = \sum_{f' \in \mathrm{RipSet}(o)} e^{-\frac{H(f')-H(f)}{T}}$$

- How to do it?

We need to divide vectors

- Vectors come with lexicographic order $\prec_{\text{lex}}$.
- Given **a** and **b**, if **0** $\prec_{\text{lex}}$ **b**, let

$$\frac{\mathbf{a}}{\mathbf{b}} := \sup\{r \in \mathbb{R} | r\mathbf{b} \prec_{\text{lex}} \mathbf{a}\}$$

- T in simulated annealing / Boltzmann distribution also a vector!
- and their quotient is a real number, or $\pm\infty$.
- Hence, $P(f \to f' | T)$ and $\frac{1}{w(f)}$ easy to compute.

We need to divide vectors

- Vectors come with lexicographic order $\prec_{\text{lex}}$.
- Given **a** and **b**, if **0** $\prec_{\text{lex}}$ **b**, let

$$\frac{\mathbf{a}}{\mathbf{b}} := \sup\{r \in \mathbb{R} | r\mathbf{b} \prec_{\text{lex}} \mathbf{a}\}$$

- T in simulated annealing / Boltzmann distribution also a vector!
- and their quotient is a real number, or $\pm\infty$.
- Hence, $P(f \to f' | T)$ and $\frac{1}{w(f)}$ easy to compute.

We need to divide vectors

- Vectors come with lexicographic order $\prec_{\text{lex}}$.
- Given **a** and **b**, if **0** $\prec_{\text{lex}}$ **b**, let

$$\frac{\mathbf{a}}{\mathbf{b}} := \sup\{r \in \mathbb{R} | r\mathbf{b} \prec_{\text{lex}} \mathbf{a}\}$$

- T in simulated annealing / Boltzmann distribution also a vector!
- and their quotient is a real number, or $\pm\infty$.
- Hence, $P(f \to f' | T)$ and $\frac{1}{w(f)}$ easy to compute.

Overview

What kind of beast is $H(f)$?

- **Total order** of $H(f)$-values.
- $H(f)$ in a **well-ordered** set:
  every non-empty subset has a least element.
- Furthermore, for the sake of learning and performance/implementation, $H(f)$ must be in an **affine space:**
  - $H(f_2) - H(f_1)$ lives in a vector space with $\prec_{\text{lex}}$,
  - and so $H(f_2) - H(f_1)$ can be divided by a vector, resulting in an "extended real number" $\in \mathbb{R} \cup \{+\infty, -\infty\}$.

What kind of beast is $H(f)$?

- **Total order** of $H(f)$-values.
- $H(f)$ in a **well-ordered** set:
  every non-empty subset has a least element.
- Furthermore, for the sake of learning and performance/implementation, $H(f)$ must be in an **affine space:**
  - $H(f_2) - H(f_1)$ lives in a vector space with $\prec_{\text{lex}}$,
  - and so $H(f_2) - H(f_1)$ can be divided by a vector, resulting in an “extended real number” $\in \mathbb{R} \cup \{+\infty, -\infty\}$.

Thank you for your attention!

Tamás Biró:
tamas[dot]biro[at]yale[dot]edu

Many thanks to:

university of groningen
faculty of arts

Amsterdam Center for Language and Communication ACLC

NWO Netherlands Organisation for Scientific Research

LUX ET VERITAS