# Research seminar week 3

## Part 1

*Tamás Biró*
*Humanities Computing*
*University of Groningen*
t.s.biro@rug.nl

# This week:

- Performance models (part 1)
- Further examples (part 2, separate slides)

# Linguistic competence

Knowledge of language in brain (Chomsky).

- Its model: a $grammar$.
- Grammar generates for each input the form that is grammatical in the language being described.
- grammatical form $=?$ correct form by native speaker

Correct form according to native speaker, not to academy.

# Approaches to linguistic competence

- Harmony Grammar (Smolensky)
- Optimality Theory (Prince and Smolensky)
- Hard constraints (Chomsky, among others)
- Principles and Parameters (Chomsky)

(Not in historical order...)

# Linguistic performance

- Utterances actually produced by native speaker (standard approach).
- Computation in brain during language production (approach of T.B.).
- Performance model: predicts the forms actually uttered/observable.

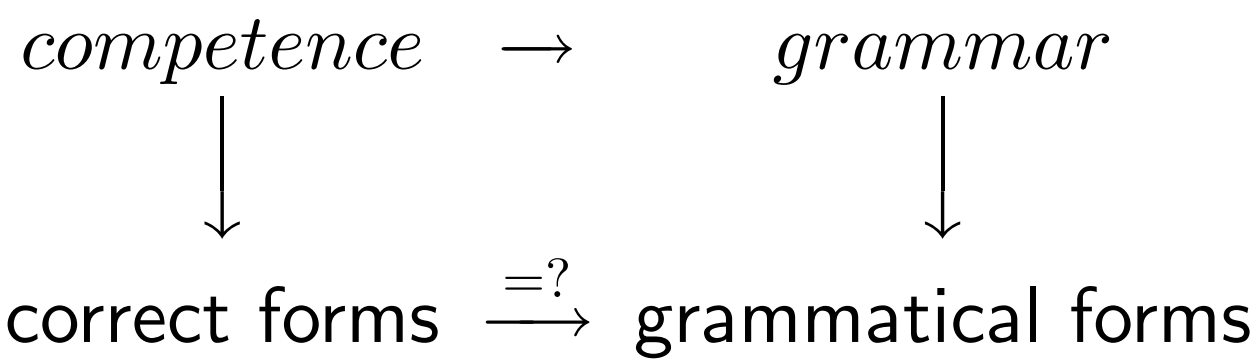
# What was a *model* for some phenomenon?

A theoretical construction (with or without maths) whose predictions mimic the observable behaviour of the phenomenon.

$$\begin{array}{ccc} phenomenon & \longrightarrow & model \\ \downarrow & & \downarrow \\ behaviour & \overset{=?}{\longrightarrow} & prediction \end{array}$$

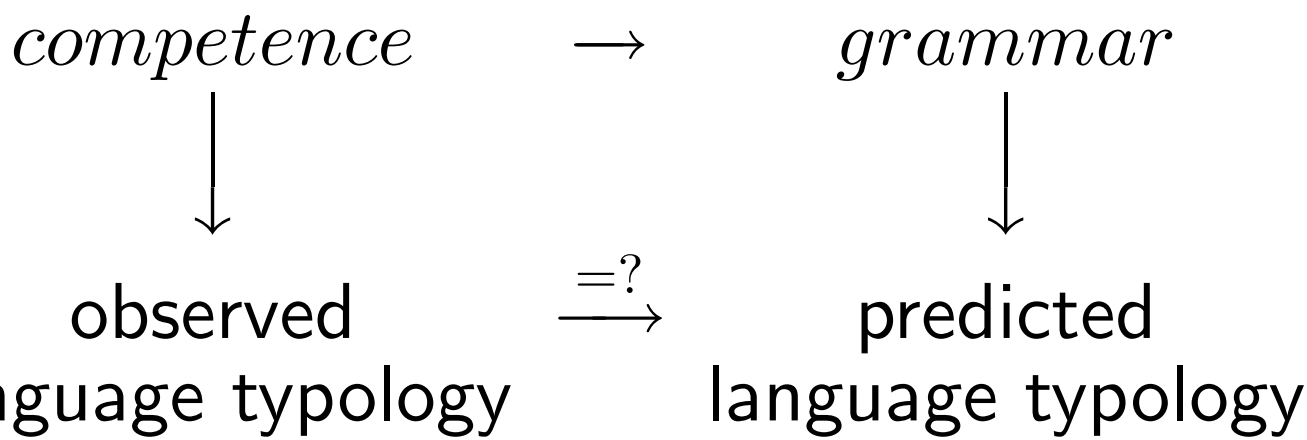
# A *model* for ling. competence

(in a single language)

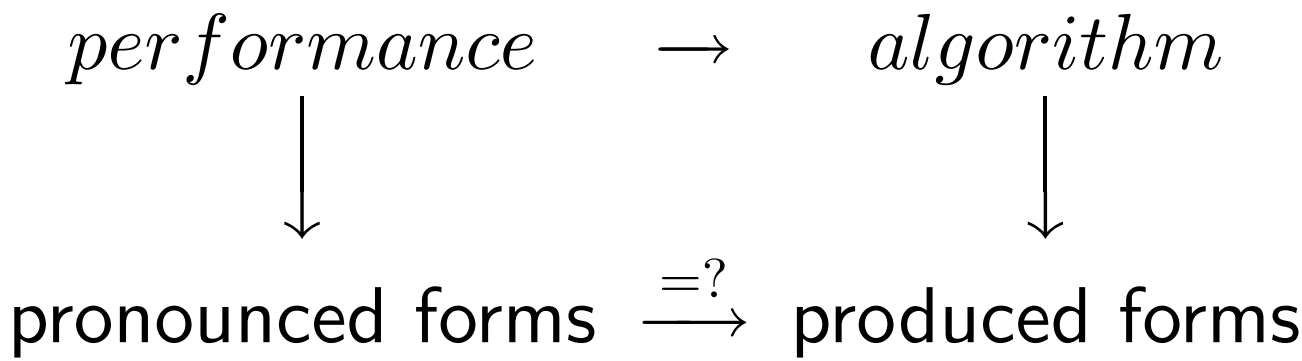


# A *model* for ling. competence

(cross-linguistic comparison)



# A *model* ling. performance



# Competence vs. performance

- Competence: a function
$$input \mapsto grammatical\ form.$$
- Performance: an algorithm realizing (implementing) this function.
Ideally, it only returns grammatical form.

# Behaviour of an algorithm

- Precision and error rate.
- Run time, complexity (time, storage space).

*What about precision, run time and complexity of human linguistic performance (computation in one's brain)?*

# Approaches to performance

- Given a compet. model (e.g., HG, OT, P&P):
- which algorithm can find the form that is predicted to be correct/grammatical by that competence model?
- Precision, run time: best vs. human-like.

Many solutions: finite-state automata, genetic algorithms, neural networks, dynamic programing, etc.

# We now focus on HG

**Task:** given candidate set $C = GEN(I)$, find maximal element(s) $c_{\text{max}}$ with respect to *target function* $H(c)$, such that

$$\forall c \in C, H(c_{\text{max}}) \geq H(c).$$

$$O(I) = \arg \max_{c \in GEN(I)} H(c).$$

# Compare competence models

- HG: optimization problem with real-valued target function.
- OT: another type of target function.
- Hard constraints, P&P: "optimization problem" with Boolean target function (good/bad).

# Ways to solve an optimization problem

- Exhaustive search
- Hill climbing 1
- Hill climbing 2
- Simulated annealing

# Sorting

Sort set $C$ (containing $n$ elements) for target function $H(c)$. Then take nr. 1.

- Precision: $100\%$, error rate: $0\%$.
- Run time (complexity) very bad: $O(n^2)$ or $O(n \cdot \log(n)))$.

See e.g. http://linux.wku.edu/~lamonml/algor/sort/sort.html

# E.g. Bubble sort: $n(n-1)/2$

```
bubbleSort(int numbers[], int array_size) {
  int i, j, temp;
  for (i = (array_size - 1); i >= 0; i--) {
    for (j = 1; j <= i; j++) {
      if (numbers[j-1] > numbers[j]) {
        temp = numbers[j-1];
        numbers[j-1] = numbers[j];
        numbers[j] = temp;
      }    } }   }
```

# Exhaustive search

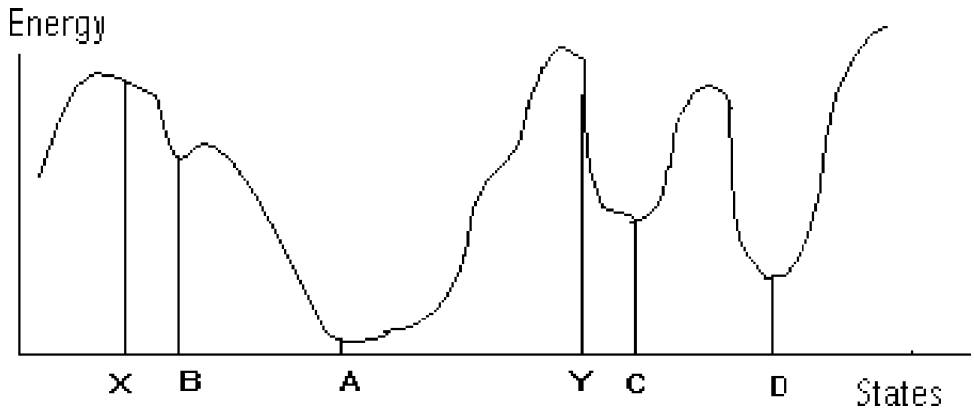
Test each possibility sequentially. Go through $C$: precision $= 1$, complexity $= n$.

```
exhaustiveSearch ( set {c_1, c_2,..., c_n} )
  Let max <-- c_1
  For i = 2 to n {
       if ( H(c_i) > H(max) ) {
            max <-- c_i
       }  }
  Return max
```

# Remarks

- Return $all$ maximal elements, or return $one$ maximal element?
- Repeat algorithm many times, to be returned some/most/all maximal elements?
- $n$ can be very large or infinite.

# Hill climbing: find global optimum




*Global minimum:* A

*Local minima:* B, C, D

*Gradient descent* from X brings to B, from Y brings to C.

# Hill climbing = gradient ascent (1)

```
w := w_init ;
Repeat
     w':= best element of set Neighbours(w);
     Delta :=  H(w') - H(w) ;
     if   Delta > 0   then   w := w' ;
     else
               do nothing
     end-if

Until stopping condition = true
Return w   # an approximation to the optimal solution
```

# Hill climbing = gradient ascent (2)

```
w := w_init ;
Repeat
     Randomly select w' from the set Neighbours(w);
     Delta :=  H(w') - H(w) ;
     if   Delta > 0   then   w := w' ;
     else
               do nothing
     end-if

Until stopping condition = true
Return w   # an approximation to the optimal solution
```

# The Simulated Annealing Algorithm

```
w := w_init ;        t := t_max  ;
Repeat
     Randomly select w' from the set Neighbours(w);
     Delta :=  H(w') - H(w) ;
     if   Delta > 0   then   w := w' ;
     else   generate random r uniformly in range (0,1);
            if   r < exp(Delta / t)   then   w := w' ;
     end-if
     t := alpha(t)                          # decrease t
Until stopping condition = true
Return w   # an approximation to the optimal solution
```

# Simulated annealing

- *Cooling schedule*: the way "temperature" `t` is decreased.
- Simulated annealing helps avoiding local maxima: increased precision.
- Slower schedule: higher precision, longer run time.
- Role of starting point `w_init`.

# Fast speech: Dutch metrical stress

| *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* | *ùit.gè.ve.ríj* | *stú.die.tòe.la.ge* | *per.fèc.tio.níst* |
| fast: *0.82* | fast: *0.65* / **0.67** | fast: *0.55* / **0.38** | fast: *0.49* / **0.1** |
| slow: *1.00* | slow: *0.97* / **0.96** | slow: *0.96* / **0.81** | slow: *0.91* / **0.2** |
| *fó.to.toe.stèl* | *ùit.ge.ve.ríj* | *stú.die.toe.là.ge* | *pèr.fec.tio.níst* |
| fast: *0.18* | fast: *0.35* / **0.33** | fast: *0.45* / **0.62** | fast: *0.39* / **0.8** |
| slow: *0.00* | slow: *0.03* / **0.04** | slow: *0.04* / **0.19** | slow: *0.07* / **0.8** |

# Recommended reading:

Tamas Biro: *When the Hothead Speaks: Simulated Annealing Optimality Theory for Dutch Fast Speech.* In T. van der Wouden, M. Poss, H. Reckman, C. Cremers (eds.): *Computational Linguistics in the Netherlands 2004*, Selected papers from the 15th CLIN meeting, LOT, Utrecht, 2005, pp. 13-28.

URL: http://roa.rutgers.edu/view.php3?id=1277.

# By next week:

- Short reports: 5 minutes/ group, showing that you have started doing something. (This is not yet one of the two presentations.)