Lecture 2a: Phonology – Word Stress

1. Inputs and outputs
2. Cross-linguistic preferences
3. OT and stress
4. The autonomy thesis
5. Autonomy breaking – the interaction of stress and syllabification

1 Inputs and outputs

The representational basis is *metrical phonology* (e.g. Liberman & Prince 1977; Halle & Vergnoud 1987; Hayes 1980, 1995). The central assumption is that stress is a rhythmic phenomenon, encoded by strong-weak relations between syllables.

In short, every prosodic word consists of patterns of alternation (termed *a foot*). Each foot contains one stressed and at most one unstressed syllable. The most common two patterns are

- *trochees* (stressed syllable on the left and at most one stressless syllable on the right), as in English, and
- *iambs* (a stressless-stressed sequence of syllables).

We use brackets to mark feet and áccents to mark prominent vowels. There are unfooted syllables (always stressless).

- **Inputs** are taken as syllabified strings of segments (motivated by morphology).
  Examples: /mi.nə.so.tə/ ; /ə.me.ri.kə/.

- **Outputs** are taken as strings which are analysed at foot-level too. We use brackets to mark feet and áccents to mark prominent vowels.
  Example: (mí.nə)(só.tə) ; ə(mé.ri)kə.

- **The Generator** can be assumed to be relatively free. Each possible input is paired with each possible output supposed the corresponding sequences of the terminal elements agree. The following are outputs generated by the input

  /ə.me.ri.kə/:

  (1) ə(mé.ri)kə

  (2) (ə.mé)(rí.kə)

  (3) ə.me(rí.kə)

  (4) (´ə.me)ri.kə

  (5) (´ə.me)ri(k´ə), ...

And these are several outputs generated by the input

/mi.nə.so.tə/:

(1) (mí.nə)(só.tə)

(2) mi.nə(só.tə)

(3) (mí.nə)so.tə

(4) mi(nə.só)tə

(5) (mi.n´ə)(só.tə)

(6) mi.nə.so.tə, ...

Notice that we drop dots if no misunderstandings are possible. For example, we write X(Y)Z instead of .X.(Y).Z.

2 Cross-linguistic preferences

The four best known common properties of stress languages:

- **The culminative property:** *Words have single prosodic peak.* Many languages impose this requirement on content words only, function words are prosodically dependent on content words.
- **The demarcative property:** *Stress tends to be placed near edges words.* Crosslinguistically favoured positions for primary word stress are (a) the initial syllable, (b) the prefinal syllable and (c) the final syllable (ranked in decreasing order of popularity among the world's languages.

- **The rhythmic property:** *Stress tends to be organized in rhythmic patterns, with strong and weak syllables spaced apart at regular intervals*. The smallest units of linguistic rhythm are metrical feet. *Trochees* are preferred. Languages may also select *iambs* .
- **Quantity-sensitivity:** *Stress prefers to fall on elements which have some intrinsic prominence*. For example, stress tends to be attracted by long vowels rather than by short ones. And stressed vowels tend to lengthen, increasing syllable weight. Mutually reinforcing relations of prominence and quantity are highly typical for stress systems.

3 OT and stress

We present a roughly simplified analysis (based on Hammond 1997) and start with the following three constraints:

Constraint corresponding to the culminative property

Words must be stressed	ROOTING

(another name for this constraint is LXWD.PRWD: grammatical words must have prosody)

Constraints corresponding to the rhythmic property

Feet are trochaic	TROCHEE
Two unfooted syllables cannot be adjacent	PARSE-SYLLABLE

Ranking for English

ROOT >> TROCH >> PARSE SYLL

Example

Input: /ə.me.ri.kə/			ROOT	TROCH	PARSE SYLL
1	☞	ə(mé.ri)kə			
2		(ə.mé)(rí.kə)		*	
3		ə.me(rí.kə)			*
4		(´ə.me)ri.kə			*
5	☞	(´ə.me)ri(k´ə)			
6	☞	(´ə.me)(rí.kə)			
7		ə.me.ri.kə	*		

The system predicts multiple (optimal) outputs. For unique solutions we have to add some more constraints. Essentially, we have to consider constraints due to the demarcative property and the property of quantity-sensitivity.

Constraint corresponding to quantity-sensitivity

Heavy syllables are stressed   WEIGHT-TO-STRESS PRINCIPLE (WSP)

Ranking for English

ROOT, WSP >> TROCH >> PARSE SYLL

Example (continued)

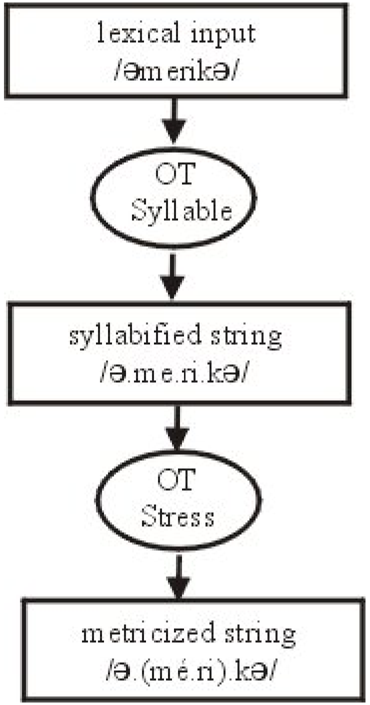
Input: / ə.me.ri.kə /		Root	WSP	Troch	Parse Syll
1 ☞	ə(mé.ri)kə				
2	(ə.mé)(rí.kə)			*	
3	ə.me(rí.kə)		*		*
4	(´ə.me)ri.kə		*		*
5	(´ə.me)ri(k´ə)		*		
6	(´ə.me)(rí.kə)		*		
7	ə.me.ri.kə	*	*		

4 The autonomy thesis

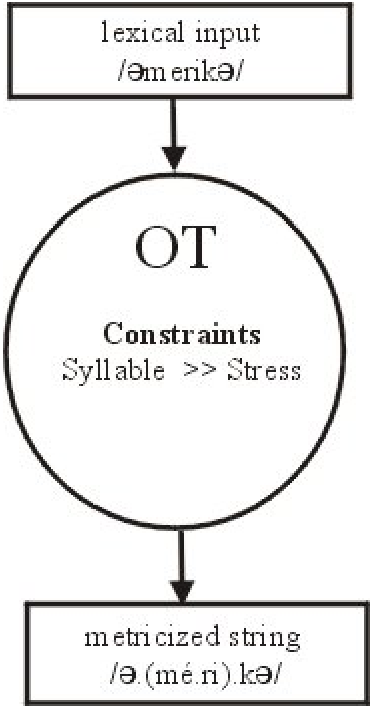
Syllabification parses a strings of segments into a sequence of substrings (called syllables). Metrical phonology adds another level of analysis and parses syllables into foots and assigns stress. In the previous section it was assumed that syllabification is independent on stress patterns. In terms of a classical cognitive architecture that means that the outputs of the system of syllabification are the inputs of the stress system. Taken this architecture, the stress system cannot have an effect on syllabification. We may refer to this hypothesis by saying

Syllabification is autonomous with regard to stress

Using OT there are two ways to formulate an autonomy thesis.
The first way, called *representational autonomy*, is a direct transportation from classical architecture into OT. The other way uses hierarchical ranking to express autonomy. It is called *dominance-based autonomy*.
Although both ways are equivalent, they are very different from a conceptual point of view and may be sources of quite different inspirations. This becomes important when violations of autonomy are envisaged.



a. representational autonomy



b. dominance-based autonomy

Representational Autonomy:

- very close to classical (rule-based) architecture.
- separation of representational units and constraints
- the outputs for one system are the inputs for the other one

Dominance-based Autonomy:

- two levels of representation only (input, output)
- no separation of representational elements necessary
- strict separation of the constraints
- the constraints of the autonomous system outrank the constraints of the dependent system

5 Autonomy breaking – the interaction of stress and syllabification

There is ample evidence that syllabification is influenced by stress, contrary to the autonomy thesis.

As a case in point consider pronunciation of /h/. This phoneme is pronounced at the beginning of words (+syllables) but not at the ends. Consequently, we can use the pronunciation of /h/ as a check for syllabification.

Now consider the pair *véhicle* – *vehícular*. In the first case /h/ isn't pronounced, in the second case it is. Consequently, our test suggests the syllabification in (i) which contrasts with standard theory (ii):

(i)	/véh.i.cle/ - /ve.hí.cu.lar/	(empirically)
(ii)	/ve.hi.cle/ - /ve.hi.cu.lar/	(standard theory)

<u>Conclusion:</u> Stress influences syllabification. Intervocalic consonants are affiliated with the syllable to its left if the following vowel is stressless.

Another example is aspiration. For example, we have the generalization that /t/ is aspirated syllable-initially but not at the ends. Consider for example the word *hotél* where the /t/ is aspirated contrasting with the word *vánity* where /t/ isn't aspirated.

(i)	/ho.tél/ - /vá.nit.y/	(empirically)
(ii)	/ho.tel/ - /va.ni.ty/	(standard theory)

The observed facts seem to obey the following constraint:

Constraint corresponding to a kind of demarcative property

Stressless medial syllables are onsetless NOONSET

Obviously, this constraint is part of the stress family and conflicts with the constraint ONSET of syllable theory. The constraint NOONSET must outrank ONSET to be effective:

$\boxed{\text{NoOnset} >> \text{Onset}}$

Autonomy breaking occurs since autonomy of syllable theory would demand that all constraints of syllable theory outrank those of the stress theory.

Interaction of stress and syllabification

Input: /vehikl/	NOONSET	ONSET	NOCODA
(vé.hikl)	*		*
☞ (véh.ikl)		*	**

Input: /vehiculχ/	NOONSET	ONSET	NOCODA
☞ ve(hí.cu)lə			
veh(í.cu)lə		*	*

Lecture 2b: Computational Aspects of OT

(based on material by J. Kuhn)

1. Computational issues
2. Some background on formal languages
3. Finite-state transducers (FSTs) and rational relations
4. Computational OT based on FSTs
5. Bidirectionality

1 Computational Issues

- **Infinity of candidate set**

Naïve evaluation algorithm for an OT system

1. construct candidates
2. apply constraints
3. pick most harmonic candidate(s)

– Since candidates can violate faithfulness constraints, the candidate set is generally infinite.
– Even with large finite candidate sets, naïve processing will get extremely costly

- **Directionality issues**

– Definition of expressive optimization is based on generation from underlying input forms – how can one decide that a given surface form is an optimal output?

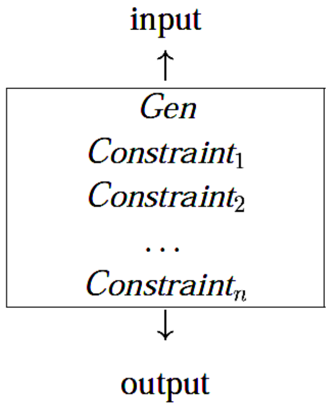
– requires processing in opposite direction to determine possible inputs (cf. robust interpretive parsing)

– this may cause additional infinity issues (even in unidirectional optimization)

- **Ways of addressing the infinity issue**

– Control candidate construction based on constraint violations dynamic (or chart-based) programming (Tesar 1995, Kuhn 2000)

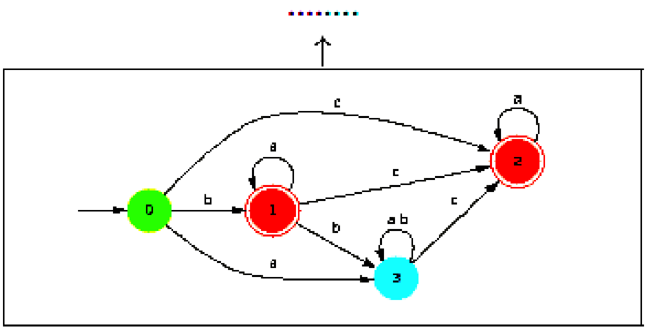
– Pre-compute the set of distinctions between relevant candidates and the respective winner (no online construction of competing candidates).
(Karttunen 1998, based on results by Frank and Satta 1998)



2 Some Background on Formal Languages

Formal languages are conceptualized as sets of strings over a given alphabet of atomic symbols (Σ).

There are at least four different ways of characterizing a formal language:

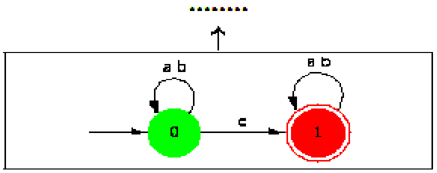
list of strings (for finite languages)	{ ϵ, a, b, aa, ab, bb, ba } (note: ϵ is the empty string)			
algebraic expression	$a^n b^n$, $n \geq 1$			
formal grammar	*non-terminals*	*terminals*	*productions*	*start symbol*
	S, A, B	a, b	$S \rightarrow A\ S\ B$	S
			$S \rightarrow \epsilon$	
			$A \rightarrow a$	
			$B \rightarrow b$	
abstract automaton				

- **Classes of formal languages (Chomsky hierarchy)**

– regular languages
– context-free languages
– context-sensitive languages
– recursively enumerable languages

The classification based on restrictions on formal grammars.
Equivalent classes follow from specific types of automata.

For instance, regular languages can equivalently be characterized in the following three ways:

regular expressions	**expressions over alphabet formed by concatenation, union and Kleene closure** **{a,b}*c{a,b}***
regular grammars	**all productions have the form A $\rightarrow$ wB or A $\rightarrow$ w, where A, B: nonterminals, w: terminal string** **S $\rightarrow$ A A $\rightarrow$ aA A $\rightarrow$ bA** **A $\rightarrow$ cC C $\rightarrow$ aC C $\rightarrow$ bC** **C $\rightarrow$ ϵ**
finite-state automata	**machine with a finite set of states and state transitions triggered by input symbols (from the alphabet) or ϵ** 

- **Important properties of regular languages:**

– closed under union, intersection, complementation

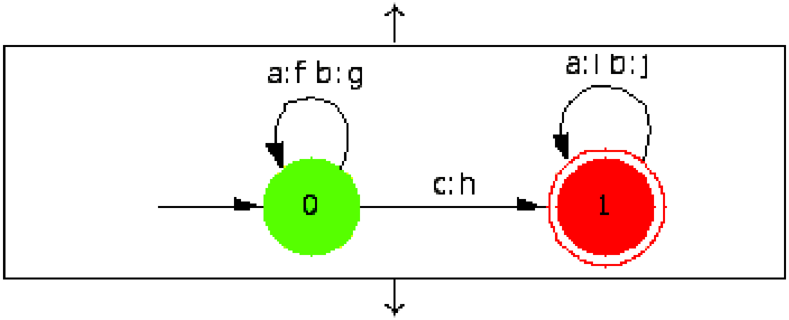
– recognizers are very efficient: linear time complexity (i.e., computation with double input length will only take twice as long)

– *Note:* Languages like $a^n b^n$ ($n \geq 1$) are not regular (but context-free)

3 OT andFinite-state transducers (FSTs) and rational relations

- a finite-state automaton with two tapes is called a finite-state transducer (FST)
- A FST specifies a relation between two regular languages (so-called rational relation)
  - state transitions are marked with two symbols a:b
  - extension of regular expression notation is used to specify transducers
  - one can view one side of the transducer as the input, which is transformed into the form(s) on the other side
  - nondeterminism may lead to several possibilities in the mapping
  - the upper and lower side can be swapped

Example 1



abcab ↔ fghij

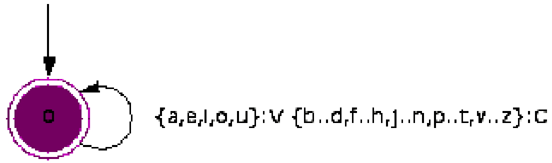
- FSTs are widely used for phonological, morphological, and "shallow" syntactic processing

Example 2

Specification:

```
{{a,e,i,o,u}:V,
{b,c,d,f,g,h,j,k,l,m,n,p,q,r,s,t,v,w,x,y,z}:C}*
```

Automaton:

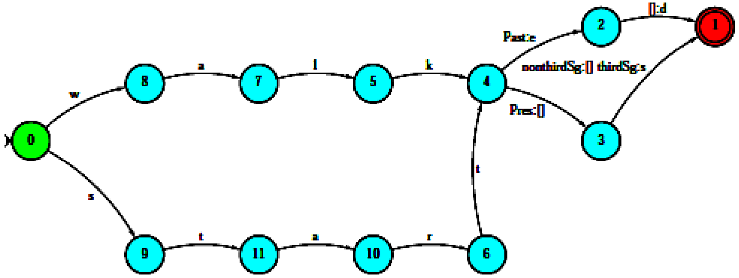


Application samples:

table	↔	CVCCV
car	↔	CVC

Example 3

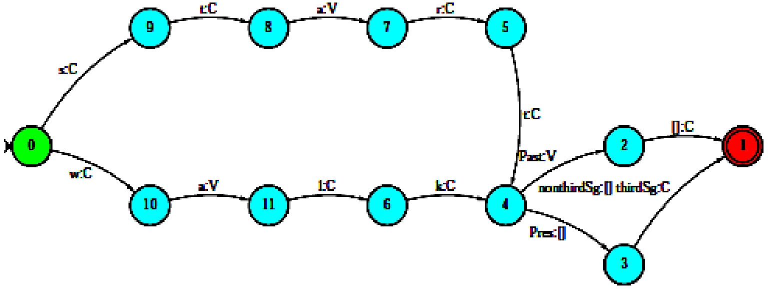
```
[{ [w,a,l,k] x [w,a,l,k],
   [s,t,a,r,t] x [s,t,a,r,t] },
 { Past x [e,d],
   [ Pres x [],
   { thirdSg x [s],
     nonthirdSg x [] } ] }      ]
```



walk Past	↔	walked
start Pres thirdSg	↔	starts
start Pres nonthirdSg	↔	start

Example 4: Composition of ex3 and ex2

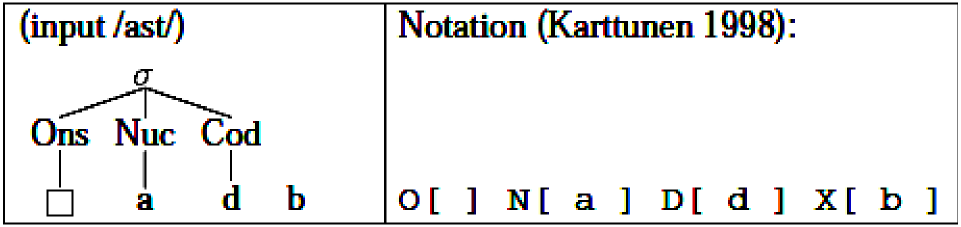
FST1 .o. FST2 maps *u* to *w* iff there is some *v* s.th. FST1 maps *u* to *v* and FST2 maps *v* to *w*.
The composition of two FSTs can be compiled into a single transducer as the following example illustrates:



walk Past	↔	CVCCVC
start Pres thirdSg	↔	CCVCCC
start Pres nonthirdSg	↔	CCVCC

4 Computational OT based on FSTs

Basic references: Frank and Satta 1998, Karttunen 1998



Gen can be defined as a transducer:

- upper side: OT input (underlying form); string of Vs and Cs.
- lower side: all possible syllabifications (including faithfulness violations)

Simplified part of the specification expression (for onset & nucleus):

```
[{[ []:'O[', {b:b,c:c,d:d ... },    []:']' ],
    []},
  [ []:'N[', {a:a,e:e,i:i,o:o,u:u}, []:']' ]
]*
```

ba ↔ a. N[] D[b] N[a] .□b.a.

b. N[] O[b] N[a] .□.ba.

c. N[] X[b] N[a] .□.<b>.a.

d. O[b] N[] N[a]

e. O[b] N[a]

f. O[b] N[a] N[]

g. O[b] N[a] D[]

h. O[] X[b] N[a]

i. X[b] N[] N[a]

j. X[b] N[a]

k. X[b] N[a] N[]

l. X[b] N[a] D[]

m. X[b] O[] N[a] ; etc.

Formalizing the constraints

Each constraint is expressed as a regular language.

Markedness

NOCODA: the language that does not contain 'D[' : `~$'D['`

ONSET: the language in which 'N[' is always preceded by 'O['...']' :
`'N['`⇒ `'O['(C)']'_`

Faithfulness

MAX-IO (No deletion.) the language that does not contain 'X[' :
`~$'X['$']'`
DEP-IO (No epenthesis.) the language in which 'O[', 'N[' and 'D['
never have ']' immediately following :
`~${'O[' ']', 'N[' ']', 'D[' ']'}`

Remark: Each simple finite-state automaton can be interpreted as a transducer (with upper and lower side identical)

What happens if we compose *Gen* and a constraint?

GEN .o. NoCoda

ba ↔	b.	N[]	O[b]		N[a]	
	c.	N[]	X[b]		N[a]	
	d.		O[b]	N[]	N[a]	
	e.		O[b]		N[a]	
	f.		O[b]		N[a]	N[]
	h.	O[]	X[b]		N[a]	
	i.		X[b]	N[]	N[a]	
	j.		X[b]		N[a]	
	k.		X[b]		N[a]	N[]
	m.		X[b]	O[]	N[a]	

GEN .o. DepIO

ba ↔	e.	O[b]	N[a]
	j.	X[b]	N[a]

GEN .o. Onset:

ba ↔	e.	O[b]		N[a]	
	g.	O[b]		N[a]	D[]
	m.	X[b]	O[]	N[a]	

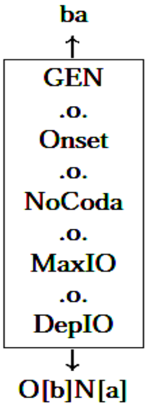
GEN .o. MaxIO:

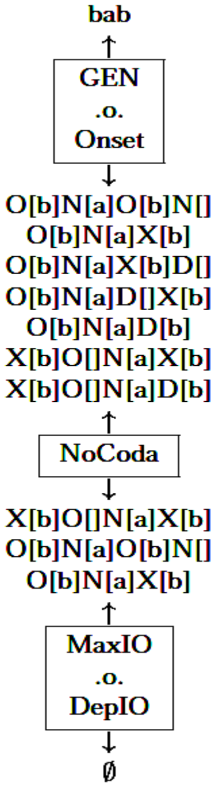
ba ↔	a.	N[]	D[b]		N[a]	
	b.	N[]	O[b]		N[a]	
	d.		O[b]	N[]	N[a]	
	e.		O[b]		N[a]	
	f.		O[b]		N[a]	N[]
	g.		O[b]		N[a]	D[]

Preliminary conclusion
Composing *Gen* and a constraint has the effect that all candidates violating the constraint are filtered out

Question
Could we compose a cascade of all the constraints to implement an OT system?
(assumed ranking: ONSET **>>** NOCODA >> MAX-IO **>>** DEP-IO)

??


ba
GEN
.o.
Onset
.o.
NoCoda
.o.
MaxIO
.o.
DepIO
O[b]N[a]


bab
GEN
.o.
Onset
O[b]N[a]O[b]N[]
O[b]N[a]X[b]
O[b]N[a]X[b]D[]
O[b]N[a]D[]X[b]
O[b]N[a]D[b]
X[b]O[]N[a]X[b]
X[b]O[]N[a]D[b]
NoCoda
X[b]O[]N[a]X[b]
O[b]N[a]O[b]N[]
O[b]N[a]X[b]
MaxIO
.o.
DepIO
∅

Preliminary conclusion

Composing *Gen* and a constraint has the effect that all candidates violating the constraint are filtered out

Question

Could we compose a cascade of all the constraints to implement an OT system?
(assumed ranking: ONSET **>>** NOCODA >> MAX-IO **>>** DEP-IO)

NO!

The problem is that this approach does not account for violability of constraints.

– Only perfect candidates will go through.
– Constraints should only be applied when at least one candidate satisfies them!

Priority Union

This operation was originally introduced as an operation for unifying two feature structures in a way that eliminates any risk of failure by stipulating that one of the two (the first one) has priority in case of a conflict:

$$Q = \left\{ \begin{array}{c} a \\ | \\ x \end{array} , \begin{array}{c} b \\ | \\ y \end{array} \right\} \qquad R = \left\{ \begin{array}{c} b \\ | \\ z \end{array} , \begin{array}{c} c \\ | \\ w \end{array} \right\}$$

$$Q\ .P.\ R = \left\{ \begin{array}{c} a \\ | \\ x \end{array} , \begin{array}{c} b \\ | \\ y \end{array} , \begin{array}{c} c \\ | \\ w \end{array} \right\}$$

Lenient composition

Using *priority union* it is possible to define a special composition operation within the FST formalism that

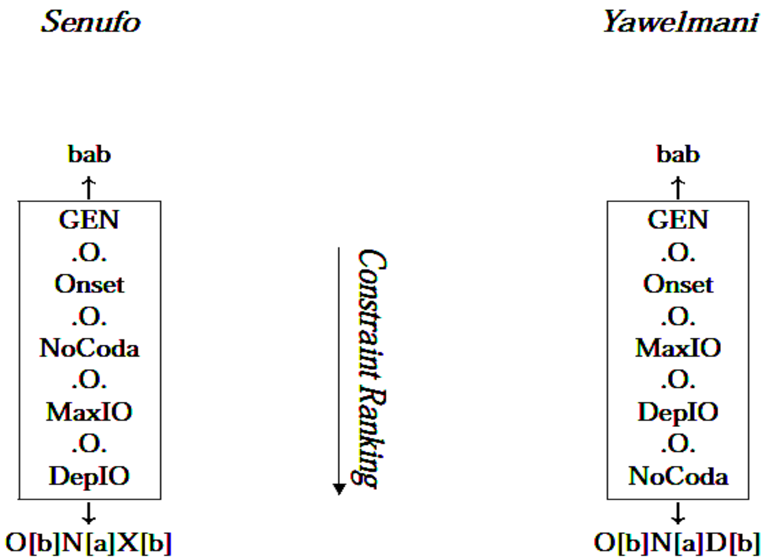
– applies a particular transducer as a filter if the resulting language is non-empty, but
– else ignores the transducer

```
R .O. C = [R .o. C] .P. R
```

Advantages

– the entire OT system is precompiled into a single transducer: a *lenient cascade*
– no runtime computation of candidates – very efficient
– compact FST: 66 (or 248) different states (Karttunen 1998 – slightly different system)

Examples of lenient cascades



.ba.<b>   .bab.

5 Bidirectionality

- Strong bidirectional optimization can be implemented based on the individual unidirectional cascades:
  - the regular languages representing the candidates after the application of the lenient cascades can be *intersected*
  - thus, strong bidirectional optimization can be expressed as a single FST

- This is not possible for weak bidirectionality (see Jäger 2000). The computational capacity of weak bidirectionality goes beyond what can be handled by FSTs.

6 Limitations of the Karttunen method

As explained in the previous slides, this method only works for constraints that can have at most a single error (Boolean constraints).

So it does not work if all surviving candidates have one error but some do not have two.

Limitations of the Karttunen method as explained in the Karttunen paper: it works only if constraints cannot have more than n errors for an n fixed per constraint

This is the best result for the FST implementation of OT as will be seen on the next slide.

And it remains fully unclear how these results work for syntax: it is not obvious syntax is FSM at all.

Why FSTs cannot capture full OT (without a bound on constraint errors) (Frank & Satta, proof by Smolensky)

The language $\{a^n b^m : n < m\}$ is not finite state recognizable

But this can be recognized in OT

GEN: $(a^n b^m), (a^n b^m)$ together with $(a^n b^m),(b^n a^m)$ is FST recognisable
Constraint set {*a}

Winners:
$(a^n b^m),(a^n b^m)$ if $n < m$
$(a^n b^m),(b^n a^m)$ if $m < n$
$(a^n b^m),(a^n b^m)$ and $(a^n b^m),(b^n a^m)$ if $n = m$

Take the right projection of the winners and intersect with $a^n b^m$ to get the target
$\{a^n b^m : n < m\}$.