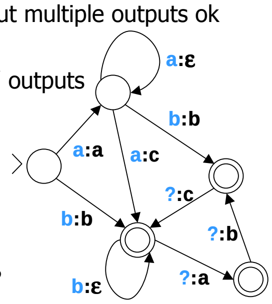
a little pre-talk review

Regular Relation (of strings)

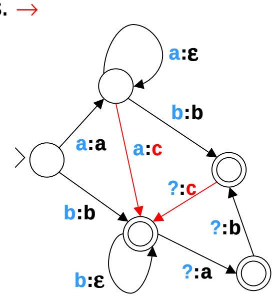
- *Relation:* like a function, but multiple outputs ok
- *Regular:* finite-state
- *Transducer:* automaton w/ outputs
- b → ? a → ?
- aaaaa → ?
- Invertible?
- Closed under composition?



a little pre-talk review

Regular Relation (of strings)

- Can weight the arcs: → vs. →
- a → {} b → {b}
- aaaaa → {ac, aca, acab, acabc}
- How to find best outputs?
  - For aaaaa?
  - For all inputs at once?



Directional Constraint Evaluation in OT

Jason Eisner
U. of Rochester

August 3, 2000 – COLING - Saarbrücken

Synopsis: Fixing OT's Power

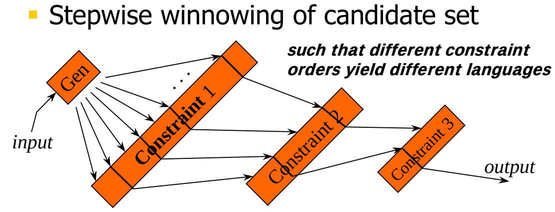
- Consensus: Phonology = regular relation
  E.g., composition of little local adjustments (= FSTs)
- Problem: Even finite-state OT is worse than that
  Global "counting" (Frank & Satta 1998)
- Problem: Phonologists want to add even more
  Try to capture iterativity by Gen. Alignment constraints
- Solution: In OT, replace counting by iterativity
  Each constraint does an iterative optimization

Outline

- Review of Optimality Theory
- The new "directional constraints" idea
- Linguistically: Fits the facts better
- Computationally: Removes excess power
- Formal stuff
  - The proposal
  - Compilation into finite-state transducers
  - Expressive power of directional constraints

What Is Optimality Theory?

- **Prince & Smolensky (1993)**
- **Alternative to stepwise derivation**
- Stepwise winnowing of candidate set



Filtering, OT-style

★★ = candidate violates constraint twice

		Constraint 1	Constraint 2	Constraint 3	Constraint 4
	Candidate A	★		★	★★★
☞	***Candidate B***		★★	★	
	Candidate C	★	★		
	Candidate D		★★★		
	Candidate E		★★	★	★
	Candidate F	★★	★★★		★

constraint would prefer A, but only allowed to break tie among B,D,E

A Troublesome Example

Input: bantodibo

		Harmony	**Faithfulness**
	ban.to.di.bo	★	
	ben.ti.do.bu	★	★★★★
	ban.ta.da.ba		★★★
☞	bon.to.do.bo		★★

"Majority assimilation" – impossible with FST - - and doesn't happen in practice!

Outline

- Review of Optimality Theory
- → The new "directional constraints" idea
- Linguistically: Fits the facts better
- Computationally: Removes excess power
- Formal stuff
  - The proposal
  - Compilation into finite-state transducers
  - Expressive power of directional constraints

An Artificial Example

Candidates have 1, 2, 3, 4 violations of **NoCoda**

		NoCoda
☞	ban.to.di.bo	★
	ban.ton.di.bo	★★
	ban.to.dim.bon	★★★
	ban.ton.dim.bon	★★★★

An Artificial Example

Add a higher-ranked constraint
This forces a tradeoff: ton vs. dim.bon

		C	**NoCoda**
	ban.to.di.bo	★	★
☞	ban.ton.di.bo		★★
	ban.to.dim.bon		★★★
	ban.ton.dim.bon		★★★★

An Artificial Example

Imagine splitting **NoCoda** into 4 syllable-specific constraints

		C	**NoCoda** σ1 σ2 σ3 σ4
	ban.to.di.bo	★	★
☞	ban.ton.di.bo		★★
	ban.to.dim.bon		★★★
	ban.ton.dim.bon		★★★★

An Artificial Example

Imagine splitting **NoCoda** into 4 syllable-specific constraints
Now ban.to.dim.bon wins - more violations but they're later

		C	NoCoda σ1	σ2	σ3	σ4
	ban.to.di.bo	★	★			
	ban.ton.di.bo		★	★		
☞	ban.to.dim.bon		★		★	★
	ban.ton.dim.bon		★	★	★	★

An Artificial Example

For "right-to-left" evaluation, reverse order (σ4 first)

		C	NoCoda σ4	σ3	σ2	σ1
	ban.to.di.bo	★				★
☞	ban.ton.di.bo				★	★
	ban.to.dim.bon		★	★		★
	ban.ton.dim.bon		★	★	★	★

Outline

- Review of Optimality Theory
- The new "directional constraints" idea
- → Linguistically: Fits the facts better
- Computationally: Removes excess power

- Formal stuff
  - The proposal
  - Compilation into finite-state transducers
  - Expressive power of directional constraints

When is Directional Different?

- The crucial configuration:

		σ1	σ2	σ3	σ4
	ban.to.di.bo	★			
	ban.ton.di.bo	★	★		
☞	ban.to.dim.bon	★		★	★

solve location conflict
by ranking locations
(sound familiar?)

- Forced location tradeoff
- Can choose where to violate, but must violate *somewhere*
- Locations aren't "orthogonal"

When is Directional Different?

- The crucial configuration:

		σ1	σ2	σ3	σ4
	ban.to.di.bo	★			
	ban.ton.di.bo	★	★		
☞	ban.to.dim.bon	★		★	★

- But if candidate 1 were available …

When is Directional Different?

- But usually locations *are* orthogonal:

		σ1	σ2	σ3	σ4
☞	ban.to.di.bo	★			
	ban.ton.di.bo	★	★		
	ban.to.dim.bon	★		★	★

- Usually, if you can satisfy σ2 and σ3 separately, you can satisfy them together
- Same winner under *either* counting or directional eval.
  (satisfies everywhere possible)

Linguistic Hypothesis

- Q: When is directional evaluation different?
- A: When something forces a location tradeoff.
- Hypothesis: Languages always resolve these cases directionally.

Test Cases for Directionality

- Prosodic groupings
  - Syllabification

    In a CV(C) language, /CVCCCV/ needs epenthesis

    V [CV.CVC.CV] Iraqi Arabic R-to-L syllabification

    V [CVC.CV.CV] Cairene Arabic L-to-R syllabification

    Analysis: **NoInsert** is evaluated R-to-L

    Analysis: **NoInsert** is evaluated L-to-R

Test Cases for Directionality

- Prosodic groupings
  - Syllabification [CV.CVC.CV] vs. [CVC.CV.CV]
  - Footing

    With binary footing, σσσσσ must have lapse

    σ(σσ)(σσ) R-to-L **Parse-σ**

    (σσ)(σσ)σ L-to-R **Parse-σ**

    (σσ)σ(σσ) unattested

Test Cases for Directionality

- Prosodic groupings
  - Syllabification [CV.CVC.CV] vs. [CVC.CV.CV]
  - Footing σ(σσ)(σσ) vs. (σσ)(σσ)σ
- Floating material
  - Lexical:
    - Tone docking ban.tó.di.bo vs. ban.to.di.bó
    - Infixation grumadwet vs. gradwumet
  - Stress "end rule" (bán.to)(di.bo) vs. (ban.to)(dí.bo)
    - OnlyStressFootHead, HaveStress » NoStress (L-R)
- Harmony and OCP effects

Generalized Alignment

- Phonology has directional phenomena
  - [CV.CVC.CV] vs. [CVC.CV.CV] - both have 1 coda, 1 V
- Directional constraints work fine
- But isn't Generalized Alignment fine too?
  - Ugly
    - Non-local; uses addition
  - Not well formalized
    - Measure "distance" to "the" target "edge"
  - Way too powerful
    - Can center tone on word, which is not possible using any system of finite-state constraints (Eisner 1997)

Outline

- Review of Optimality Theory
- The new "directional constraints" idea
- Linguistically: Fits the facts better
- → Computationally: Removes excess power

- Formal stuff
  - The proposal
  - Compilation into finite-state transducers
  - Expressive power of directional constraints

Computational Motivation

- Directionality not just a substitute for GA
- Also a substitute for counting
- Frank & Satta 1998:

  OTFS > FST

  (Finite-state OT is *more powerful* than finite-state transduction)

Why OTFS > FST?

- It matters that OT can count
- **HeightHarmony » HeightFaithfulness**
- Input: to.tu.to.to.tu
- Output: to.to.to.to.to
  vs. tu.tu.tu.tu.tu

can both be implemented by weighted FSAs

prefer candidate with fewer faithfulness violations

- Majority assimilation (Baković 1999, Lombardi 1999)
- Beyond FST power - fortunately, unattested

Why Is OT > FST a Problem?

- Consensus: Phonology = regular relation
  - OT supposed to offer elegance, not power
- FSTs have many benefits!
  - Generation in linear time (with no grammar constant)
  - Comprehension likewise (cf. <u>no</u> known OTFS algorithm)
    - Invert the FST
    - Apply in parallel to weighted speech lattice
    - Intersect with lexicon
  - Compute difference between 2 grammars

Making OT=FST: Proposals

- Approximate by bounded constraints
  - Frank & Satta 1998, Karttunen 1998
  - Allow only up to 10 violations of **NoCoda**
  - Yields huge FSTs - cost of missing the generalization
- Another approximation
  - Gerdemann & van Noord 2000
  - Exact if location tradeoffs are between close locations
- <u>Allow directional and/or bounded constraints only</u>
  - Directional **NoCoda** correctly disprefers *all* codas
  - Handle location tradeoffs by ranking locations
  - Treats counting as a bug, not a feature to approximate

Outline

- Review of Optimality Theory
- The new "directional constraints" idea
- Linguistically: Fits the facts better
- Computationally: Removes excess power
- Formal stuff
  - → The proposal
  - Compilation into finite-state transducers
  - Expressive power of directional constraints

Tuples

- Violation levels aren't integers like ★★★
- They're integer *tuples*, ordered lexicographically

		NoCoda σ1	σ2	σ3	σ4
	ban.ton.di.bo	1	1	0	0
☞	ban.to.dim.bon	1	0	1	1
	ban.ton.dim.bon	1	1	1	1

Tuples

- Violation levels aren't integers like ★★★
- They're integer *tuples*, ordered lexicographically
- But what about candidates with 5 syllables?
  - And syllables aren't fine-grained enough in general

		NoCoda			
		σ1	σ2	σ3	σ4
	ban.ton.di.bo	1	1	0	0
☞	ban.to.dim.bon	1	0	1	1
	ban.ton.dim.bon	1	1	1	1

Alignment to Input

- Split by input symbols, not syllables
- Tuple length = input string length + 1

Input:		b	a	n	t	o	d	i	b	o
Output:		b	a	n	t	o n	d	i	b	o
	0	0	0	1	0	1	0	0	0	0

For this input (length 9),
NoCoda assigns each output candidate a 10-tuple
Possible because output is aligned with the input
So each output violation associated with an input position

Alignment to Input

- Split by input symbols, not syllables
- Tuple length = input length + 1, for all outputs

Input:		b	a	n	t	o	d	i	b	o
Output:		b	a	n	t	o n	d	i	b	o
	0	0	0	1	0	1	0	0	0	0
Output:		b	a	n	t	o	d	i m	b	o n
	0	0	0	1	0	0	0	1	0	1

Alignment to Input

- Split by input symbols, not syllables
- Tuple length = input length + 1, for all outputs

Input:		b	a	n	t	o	d	i	b	o
Output:		b	a	n	t	o n	d	i	b	o
	0	0	0	1	0	1	0	0	0	0
Output:		b	a	n	t	o	d	i m	b	o n
	0	0	0	1	0	0	0	1	0	1
Output:	i	b	a	n	t	o n	d	i mtim	b	o n n n
	0	0	0	1	0	0 1	0	2	0	3

Alignment to Input

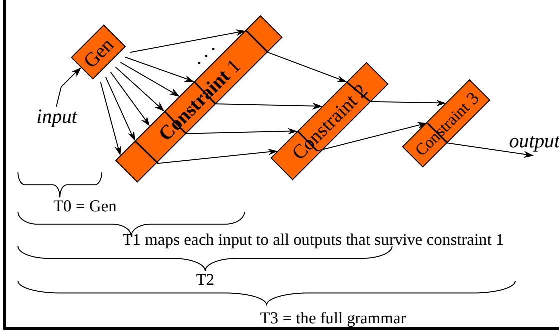
- Split by input symbols, not syllables
- Tuple length = input length + 1, for all outputs

Input:		b	a	n	t	o	d	i	b	o
Output:		b	a	n	t	o n	d	i	b	o
	0	0	0	1	0	1	0	0	0	0
Output:										
Output:	i	b	a	n	t	o n	d	i mtim	b	o n n n
	0	0	0	1	0	0 1	0	2	0	3

does *not* count as "postponing" n
so this candidate doesn't win (thanks to alignment)

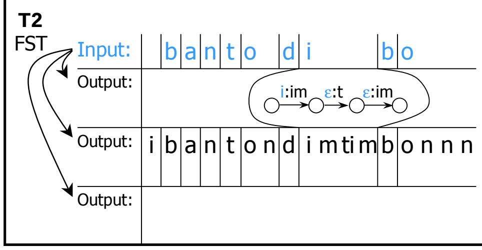
unbounded

Finite-State Approach



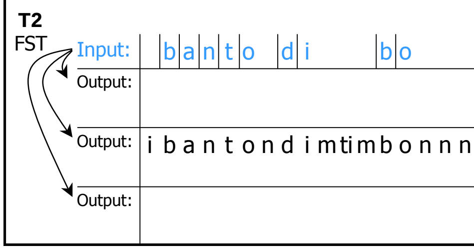
Finite-State Approach

- FST maps each input to set of outputs
  (nondeterministic mapping)
- The transducer gives an alignment



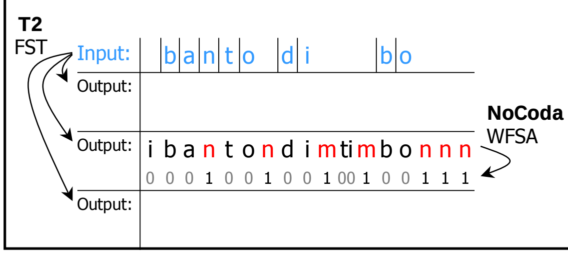
Finite-State Machines

- FST maps each input to set of outputs



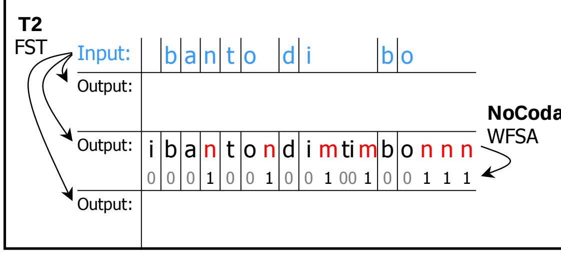
Finite-State Machines

- FST maps each input to set of aligned outputs
- Constraint is a weighted FSA that reads candidate



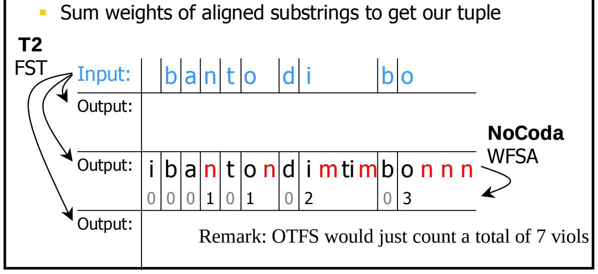
Finite-State Machines

- FST maps input to aligned candidates (nondeterm.)
- Constraint is a weighted FSA that reads candidate



Finite-State Machines

- FST maps input to aligned candidates (nondeterm.)
- Constraint is a weighted FSA that reads candidate
- Sum weights of aligned substrings to get our tuple



Similar Work

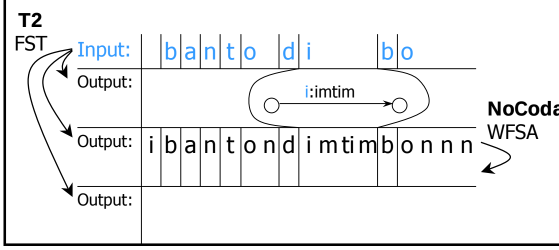
- Bounded Local Optimization
  - Walther 1998, 1999 (for DP)
  - Trommer 1998, 1999 (for OT)
  - An independent proposal
    - Motivated by directional syllabification
  - Greedy pruning of a candidate-set FSA
    - Violations with different prefixes are incomparable
    - No alignment, so insertion can postpone violations
    - No ability to handle multiple inputs at once (FST)

Outline

- Review of Optimality Theory
- The new "directional constraints" idea
- Linguistically: Fits the facts better
- Computationally: Removes excess power
- Formal stuff
  - The proposal
  - Compilation into finite-state transducers
  - Expressive power of directional constraints

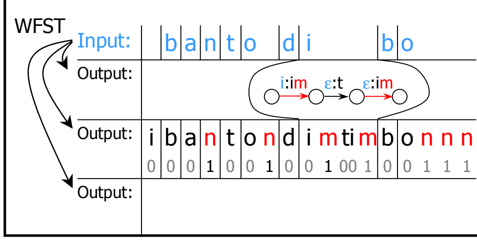
The Construction

- Our job is to construct T3 - a "filtered" version of T2
  - First compose T2 with **NoCoda** …



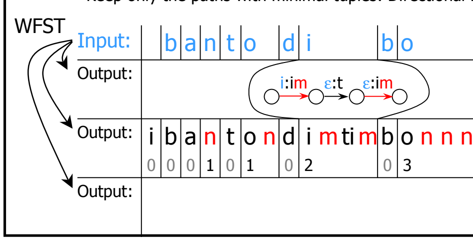
The Construction

- Our job is to construct T3 - a "filtered" version of T2
  - First compose T2 with **NoCoda** to get a weighted FST



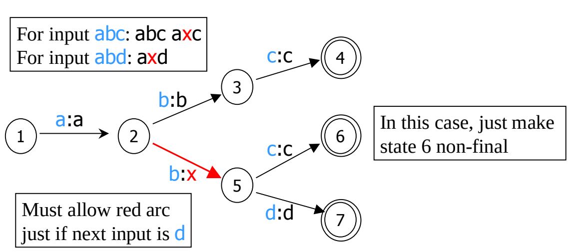
The Construction

- Our job is to construct T3 - a "filtered" version of T2
  - First compose T2 with **NoCoda** to get a weighted FST
  - Now prune this weighted FST to obtain T3
  - Keep only the paths with minimal tuples: Directional Best Paths



Directional Best Paths (sketch)

- Handle all inputs simultaneously!
- Must keep best outputs for each input: at least 1.



Directional Best Paths (sketch)

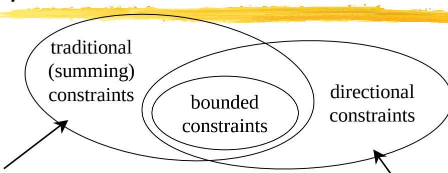
- Must pursue counterfactuals
- Recall determinization (2^n states)
  - DFA simulates a parallel traverser of the NDFA
  - "What states could I be in, given input so far?"
- Simulate a neurotic traverser of the WFST
  - "If I had taken a cheaper (greedier) path on the input so far, what states could I be in right now?"
  - Shouldn't proceed to state *q* if there was a cheaper path to *q* on same input
  - Shouldn't terminate in state *q* if there was a cheaper terminating path (perhaps to state *r*) on same input
  - 3^n states: track statesets for equal and cheaper paths

Outline

- Review of Optimality Theory
- The new "directional constraints" idea
- Linguistically: Fits the facts better
- Computationally: Removes excess power

- Formal stuff
  - The proposal
  - Compilation into finite-state transducers
  - → Expressive power of directional constraints

Expressive Power



a traditional constraint with > FST power can't be replaced by any system of directional constraints

a directional constraint making exponentially many distinctions can't be replaced by any system of trad. finite-state constraints

***b** (L-R) sorts $\{a,b\}^n$ alphabetically

Future Work

- Further empirical support?
- Examples where 1 early violation trades against 2 late violations of the same constraint?
- How do directional constraints change the style of analysis?
- How to formulate constraint families? (They must specify precisely where violations fall.)

An Old Slide (1997)

Same power as Primitive OT (formal linguistic proposal of Eisner 1997)

FST < OTFS < OTFS + GA

Should we pare OT back to this level?

Hard to imagine making it any simpler than Primitive OT.

Should we beef OT up to this level, by allowing GA?

Ugly mechanisms like GA weren't needed before OT.

The New Idea (2000)

Same power as Primitive OT (formal linguistic proposal of Eisner 1997)

directionality

FST = OTFS = OTFS + ~~GA~~

~~(summation)~~ directionality

Should we pare OT back to this level?

~~***Hard to imagine making it any simpler than Primitive OT.***~~

Should we beef OT up to this level, by allowing GA?

Ugly mechanisms like GA weren't needed before OT.