Inducing Phonological Rules: Perspectives from Bayesian Program Learning

Kevin Ellis and Timothy O'Donnell and Joshua B. Tenenbaum and Armando Solar-Lezama

MIT

2017

A language learning experiment

xau

man

kwaj

çin

A language learning experiment

xau	xauxaudə
man	manmandə
kwaj	kwajkwajdə
çin	çinçində

A language learning experiment

xau	xauxaudə
man	manmandə
kwaj	kwajkwajdə
çin	çinçində
lej	

A language learning experiment

xau	xauxaudə
man	manmandə
kwaj	kwajkwajdə
çin	çinçində
lej	lejlejdə

A language learning experiment

xau	xauxaudə
man	manmandə
kwaj	kwajkwajdə
çin	çinçində
lej	lejlejdə

$$\mathtt{A} \longrightarrow \mathtt{AA} + \mathtt{d\text{ə}}$$

A language learning experiment

dom

kot

lut

vus

wuk

bur

A language learning experiment

dom	domi
kot	koti
lut	lodi
vus	vozi
wuk	wugi
bur	bori

A language learning experiment

dom	domi
kot	koti
lut	lodi
vus	vozi
wuk	wugi
bur	bori
	rogi

A language learning experiment

dom	domi
kot	koti
lut	lodi
vus	vozi
wuk	wugi
bur	bori
ruk	rogi

A language learning experiment

dom	domi
kot	koti
lut	lodi
vus	vozi
wuk	wugi
bur	bori
ruk	rogi

$\mathtt{A} \rightarrow \mathtt{A}+/\mathrm{i}/$

o$\rightarrow$u/ _ [-nasal +voice]#

[-sonorant]$\rightarrow$[-voice] / _ #

ABA (same/different/same)

wofewo

lovilo

fimufi

ABA (same/different/same)

wofewo

lovilo

fimufi

$$\varnothing \rightarrow \sigma_i \ / \ \sigma_i \sigma \ \_$$

$$/\text{wofe}/ \rightarrow [\text{wofewo}]$$

Motivation

Problem: Understand the principles that underlie linguistic generalization.

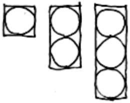
Approach: Triangulate these inductive principles with a wide variety of problems/data sets

A (reverse) engineering problem

How is it possible to learn a large number of diverse natural and artificial grammars from relatively small data sets?

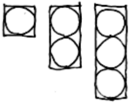
Bayesian Program Learning (BPL)

Bayesian Program Learning (BPL)



```
for (i < 3)
 rectangle(3*i,-2*i+4,
           3*i+2,6)
 for (j < i + 1)
  circle(3*i+1,-2*j+5)
```

Bayesian Program Learning (BPL)

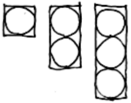


```
for (i < 3)
 rectangle(3*i,-2*i+4,
           3*i+2,6)
 for (j < i + 1)
  circle(3*i+1,-2*j+5)
```

Input	Output
“1/21/2001”	“01”

`substr(pos('0',-1),-1)`	“last 0 til end”
`const('01')`	“output 01”
`substr(-2,-1)`	“take last two”

Bayesian Program Learning (BPL)



```
for (i < 3)
 rectangle(3*i,-2*i+4,
           3*i+2,6)
 for (j < i + 1)
  circle(3*i+1,-2*j+5)
```

Input	Output
"1/21/2001"	"01"

`substr(pos('0',-1),-1)`	"last 0 til end"
`const('01')`	"output 01"
`substr(-2,-1)`	"take last two"

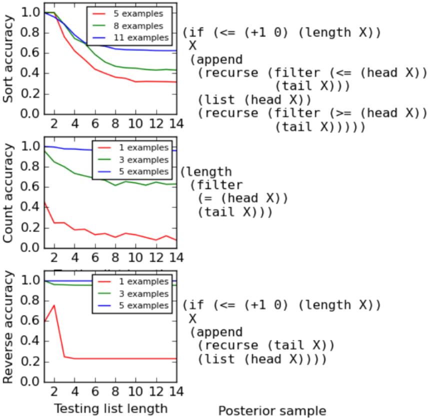
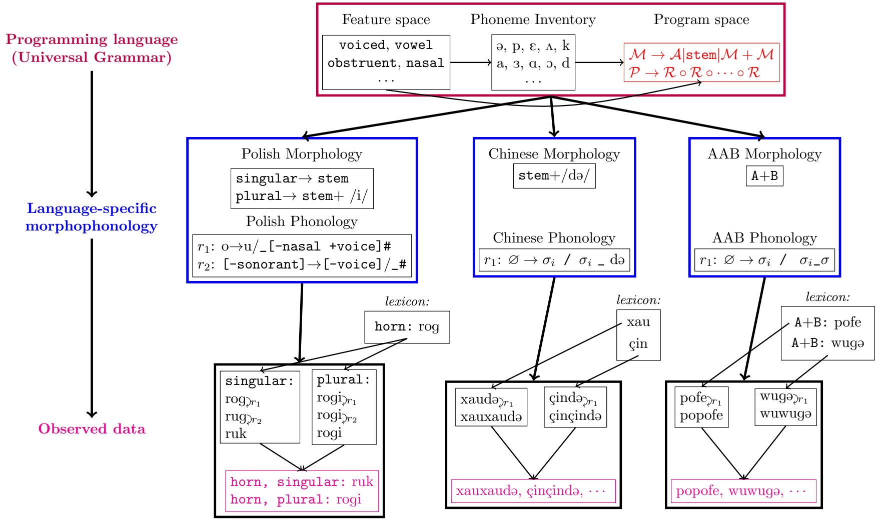


Figure 9: Learning to manipulate lists. Trained on lists of length ≤ 3; tested on lists of length ≤ 14.

Talk roadmap

- Bayesian program learning (BPL) for phonology
- Artificial grammar learning: using BPL as a tool for studying simplicity trade-offs
- Phonological rules in natural language: BPL as a tool for explaining a breadth of phenomena
- Simplicity metrics and universal grammar
- A problem with minimum description length as a simplicity metric

Bayesian Program Learning (BPL) for phonology



What makes a good model?

Probable$\Longleftrightarrow$low description length$\Longleftrightarrow$simple

$$\text{descriptionLength}(\mathit{randomEvent}) = -\log \mathbb{P}[\mathit{randomEvent}]$$

What makes a good model?

Probable$\iff$low description length$\iff$simple

$$\text{descriptionLength}(\mathit{randomEvent}) = -\log \mathbb{P}[\mathit{randomEvent}]$$

$$\begin{aligned}\text{descriptionLength}(\text{model}; \text{data}) &= \text{descriptionLength}(\text{model}) \\ &+ \sum_{x \in \text{data}} \text{descriptionLength}(x; \text{model})\end{aligned}$$

What makes a good model?

Probable$\Longleftrightarrow$low description length$\Longleftrightarrow$simple

$$\text{descriptionLength}(\mathit{randomEvent}) = -\log \mathbb{P}[\mathit{randomEvent}]$$

$$\begin{aligned}\text{descriptionLength}(\text{model}; \text{data}) &= \text{descriptionLength}(\text{model}) \\ &+ \sum_{x \in \text{data}} \text{descriptionLength}(x; \text{model})\end{aligned}$$

$$\text{descriptionLength}(\text{model}) \sim \text{program size}$$

What makes a good model?

Probable$\Longleftrightarrow$low description length$\Longleftrightarrow$simple

$$\text{descriptionLength}(\textit{randomEvent}) = -\log \mathbb{P}[\textit{randomEvent}]$$

$$\begin{aligned}\text{descriptionLength}(\text{model}; \text{data}) &= \text{descriptionLength}(\text{model}) \\ &+ \sum_{x \in \text{data}} \text{descriptionLength}(x; \text{model})\end{aligned}$$

$$\text{descriptionLength}(\text{model}) \sim \text{program size}$$

$$\text{descriptionLength}(x; \text{model}) = \text{size of } x\text{'s UR in model}$$

$$\text{descriptionLength}([\text{pofepo}]; \text{ABA}) = \text{len}(/\text{pofe}/) = 4$$

(see Richard Futrell's talk)

What makes a good model?

Probable$\Longleftrightarrow$low description length$\Longleftrightarrow$simple

$$\text{descriptionLength}(\textit{randomEvent}) = -\log \mathbb{P}[\textit{randomEvent}]$$

$$\begin{aligned}\text{descriptionLength}(\text{model}; \text{data}) &= \text{descriptionLength}(\text{model}) \\ &+ \sum_{x \in \text{data}} \text{descriptionLength}(x; \text{model})\end{aligned}$$

$$\text{descriptionLength}(\text{model}) \sim \text{program size}$$

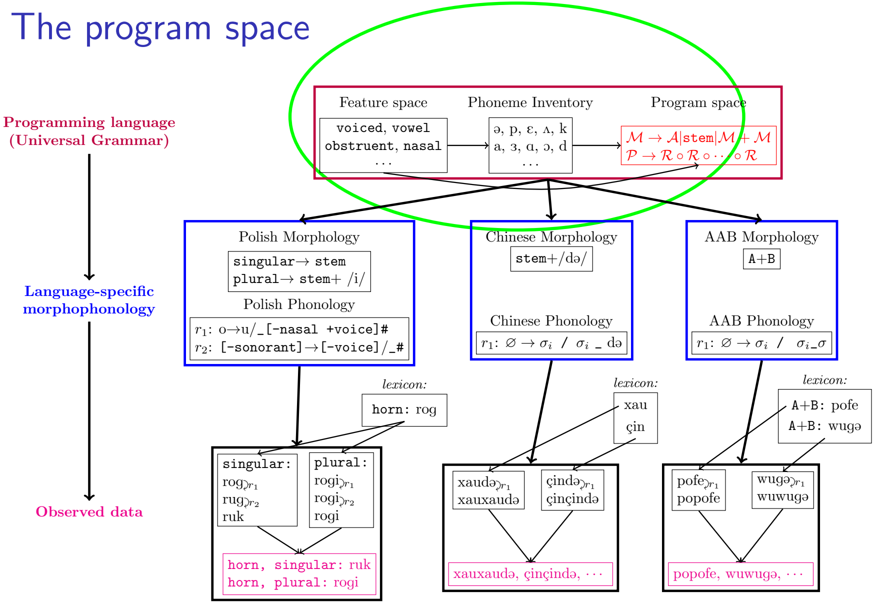
$$\text{descriptionLength}(x; \text{model}) = \text{size of } x\text{'s UR in model}$$

$$\text{descriptionLength}([\text{pofepo}]; \text{ABA}) = \text{len}(/\text{pofe}/) = 4$$

(see Richard Futrell's talk)

Bayesian Program Learning framed as compression
Old idea from Solomonoff

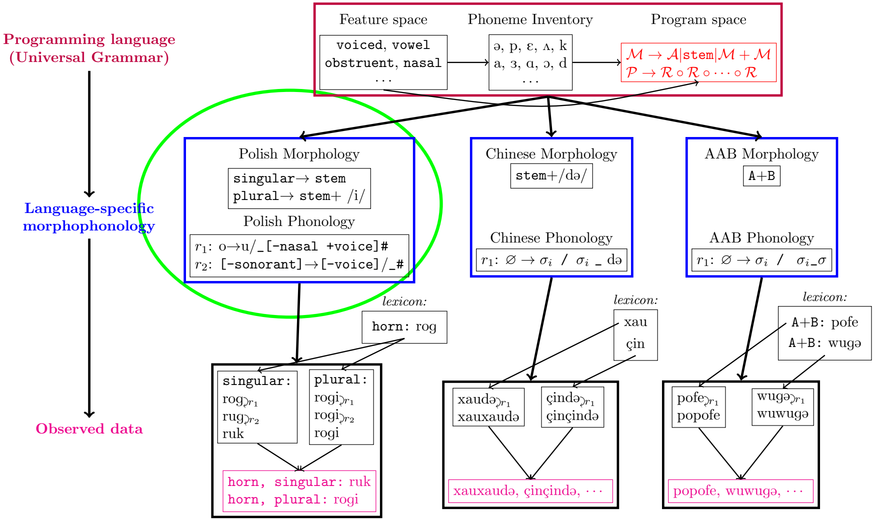
The program space



The program space: SPE-style context-sensitive rewrites

Grammar rule	English description
$\mathcal{P} \rightarrow \mathcal{R} \circ \mathcal{R} \circ \cdots \circ \mathcal{R}$	**P**honology is compositions of **r**ewrites
$\mathcal{R} \rightarrow \mathcal{F} \longrightarrow \mathcal{C}/\mathcal{T}\_\mathcal{T}$	Rewrite **f**ocus to **c**hange between **t**riggers
$\mathcal{T} \rightarrow \#\mathcal{T}'\|\mathcal{T}'$	**T**riggers optionally match end of string, $\#$
$\mathcal{T}' \rightarrow \epsilon\|\mathcal{X}\mathcal{T}'\|\mathcal{X}^*\mathcal{T}'$	**T**riggers are sequences of matrices $\mathcal{X}$
$\mathcal{X} \rightarrow \mathsf{a}\|t\|s\|\cdots$	Matrices can be constant phonemes
$\mathcal{X} \rightarrow [\pm\mathcal{E}\pm\mathcal{E}\cdots\pm\mathcal{E}]$	Matrices check features $\mathcal{E}$
$\mathcal{E} \rightarrow \text{voice}\|\text{nasal}\|\cdots$	Standard phonological f**e**atures
$\mathcal{F} \rightarrow \mathcal{X}$	Focus can be a feature matrix
$\mathcal{F} \rightarrow \mathbb{Z}$	Focus can be one of the triggers (copies it)
$\mathcal{F} \rightarrow \varnothing$	Insertion rule
$\mathcal{C} \rightarrow \mathcal{X}$	Structural change can be a feature matrix
$\mathcal{C} \rightarrow \varnothing$	Deletion rule
$\mathcal{C} \rightarrow \mathbb{Z}$	Structural change constrained to match a triggering feature matrix

The programs



The programs

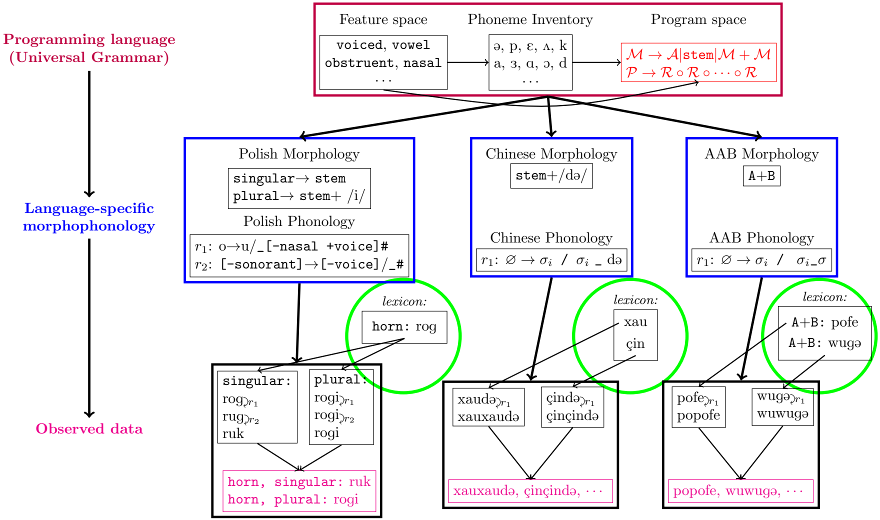
- Morphology: Simple concatenative rules combining underlying forms of morphemes based on morphological function.

SING: # + rog + #
PLURAL: # + rog + i + #

- Phonology: Ordered rules that transform resulting phone sequences.

$o \rightarrow u / \_ \text{[-nasal +voice]\#}$
$\text{obstruent} \rightarrow \text{[-voice]} / \_\#$

The lexicon



Programming language
(Universal Grammar)
Language-specific
morphophonology
Observed data
Feature space
voiced, vowel
obstruent, nasal
...
Phoneme Inventory
ə, p, ɛ, ʌ, k
a, ʒ, ɑ, ɔ, d
...
Program space
$\mathcal{M} \to \mathcal{A}|\mathtt{stem}|\mathcal{M}+\mathcal{M}$
$\mathcal{P} \to \mathcal{R} \circ \mathcal{R} \circ \cdots \circ \mathcal{R}$
Polish Morphology
singular→ stem
plural→ stem+ /i/
Polish Phonology
r_1: o→u/_[-nasal +voice]#
r_2: [-sonorant]→[-voice]/_#
Chinese Morphology
stem+/də/
Chinese Phonology
r_1: $\varnothing \to \sigma_i$ / σ_i _ də
AAB Morphology
A+B
AAB Phonology
r_1: $\varnothing \to \sigma_i$ / σ_i_σ
lexicon:
horn: rog
lexicon:
xau
çin
lexicon:
A+B: pofe
A+B: wugə
singular:
rog
rug
ruk
plural:
rogi
rogi
rogi
horn, singular: ruk
horn, plural: rogi
xaudə
xauxaudə
çində
çinçində
xauxaudə, çinçində, ⋯
pofe
popofe
wugə
wuwugə
popofe, wuwugə, ⋯

The lexicon

Inventory of stems ("underlying representations")

Polish:
rog
klub
dvon
...

AAB:
pofe
wuga
...

Shorter stem $\Longleftrightarrow$ Better compression $\Longleftrightarrow$ Higher likelihood of data

But how do you find any good programs in the first place?

The search problem

program synthesis techniques from Armando Solar-Lezama

SAT/SMT solving:

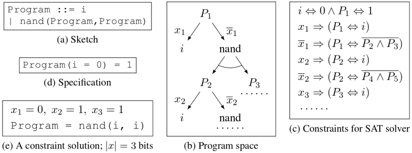


Figure 2: Synthesizing a program via sketching and constraint solving. Typewriter font refers to pieces of programs or sketches, while math font refers to pieces of a constraint satisfaction problem. The variable i is the program input.

Sketch:

```
generator int rec(int x, int y, int z){
    int t = ??;
    if(t == 0){return x;}
    if(t == 1){return y;}
    if(t == 2){return z;}

    int a = rec(x,y,z);
    int b = rec(x,y,z);

    if(t == 3){return a * b;}
    if(t == 4){return a + b;}
    if(t == 5){return a - b;}
}
harness void sketch( int x, int y, int z ){
    assert rec(x,y, z) == (x + x) * (y - z);
}
```

The search problem

program synthesis techniques from Armando Solar-Lezama

SAT/SMT solving:

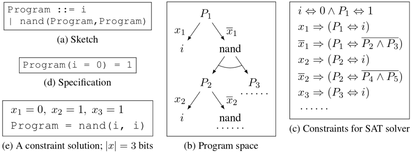


Figure 2: Synthesizing a program via sketching and constraint solving. `Typewriter font` refers to pieces of programs or sketches, while math font refers to pieces of a constraint satisfaction problem. The variable `i` is the program input.

Sketch:

```
generator int rec(int x, int y, int z){
    int t = ??;
    if(t == 0){return x;}
    if(t == 1){return y;}
    if(t == 2){return z;}

    int a = rec(x,y,z);
    int b = rec(x,y,z);

    if(t == 3){return a * b;}
    if(t == 4){return a + b;}
    if(t == 5){return a - b;}
}
harness void sketch( int x, int y, int z ){
    assert rec(x,y, z) == (x + x) * (y - z);
}
```

✓ Guarantee: Exact optimization

✗ No guarantee: runtime

The search problem

program synthesis techniques from Armando Solar-Lezama

SAT/SMT solving:

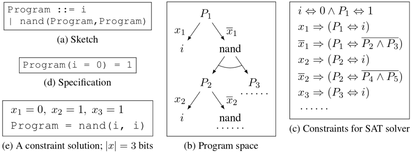


Figure 2: Synthesizing a program via sketching and constraint solving. `Typewriter font` refers to pieces of programs or sketches, while math font refers to pieces of a constraint satisfaction problem. The variable `i` is the program input.

Sketch:

```
generator int rec(int x, int y, int z){
    int t = ??;
    if(t == 0){return x;}
    if(t == 1){return y;}
    if(t == 2){return z;}

    int a = rec(x,y,z);
    int b = rec(x,y,z);

    if(t == 3){return a * b;}
    if(t == 4){return a + b;}
    if(t == 5){return a - b;}
}
harness void sketch( int x, int y, int z ){
    assert rec(x,y, z) == (x + x) * (y - z);
}
```

✓ Guarantee: Exact optimization

✗ No guarantee: runtime

2 rules, 1 inflection:

$\geq (10^{18}\text{rules})^2 \times (10^8\text{morphologies}) = 10^{42}\text{models}$

Artificial Grammar Learning

Widely studied.

Fundamental hypothesis: AGL engages some shared resources with first language acquisition

- ABA (same/different/same): wofewo, pikæpi, gugagu
- ABB (different/same/same): wowofe, pipikæ, guguga
- Pig Latin
- ...

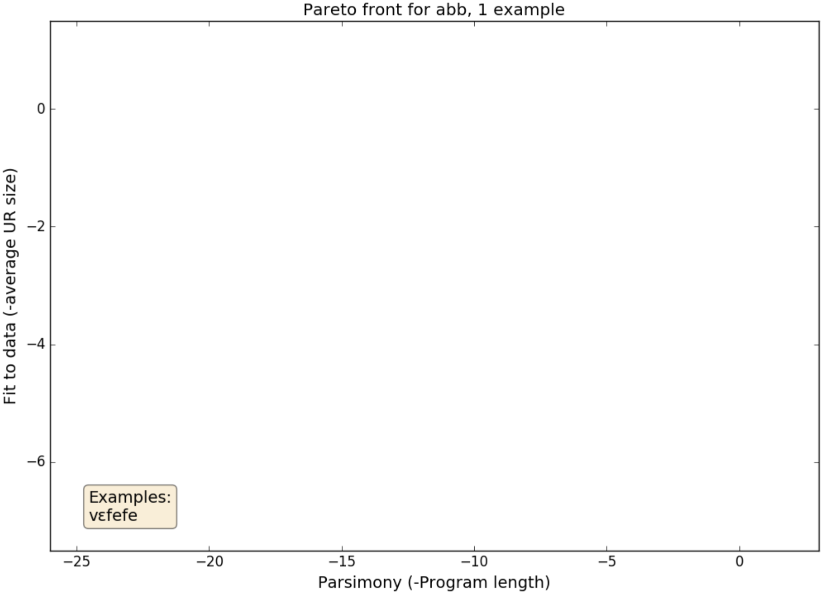
Artificial Grammar Learning

Grammar	Example input to learner	MAP grammar	Natural language analogues
ABA (same/different/same) Marcus et al. 1999.	wofewo lovilo fimufi	$\varnothing \rightarrow \sigma_i$ / #_$\sigma\sigma_i$	Reduplication (eg, Tagalog)
ABB (different/same/same) Marcus et al. 1999.	wofefe lovivi fimumu	$\varnothing \rightarrow \sigma_i$ / σ_i_#	Reduplication (eg, Tagalog)
ABx (different/different/constant)	wofeka lovika fimuka	$\varnothing \rightarrow x$ / _#	concatenative morphology
AAx (same/same/constant) Gerken 2006.	wowoka loloka fifika	$\varnothing \rightarrow \sigma_i$ / #_σ_i $\varnothing \rightarrow x$ / _#	reduplication concatenative morphology
AxA (same/constant/same) Gerken 2006.	wokawo lokalo fikafi	$\varnothing \rightarrow$ a / # _ $\varnothing \rightarrow$ k / # _ $\varnothing \rightarrow \sigma_i$ / # _ σ σ_i	Infixing Reduplication
Pig Latin	pɪg→ɪgpe latɪn→atɪle æsk→æske	$\varnothing \rightarrow C_i$ / #C_i [ ]* _ # $\varnothing \rightarrow$ e / _ # $C \rightarrow \varnothing$ / # _	

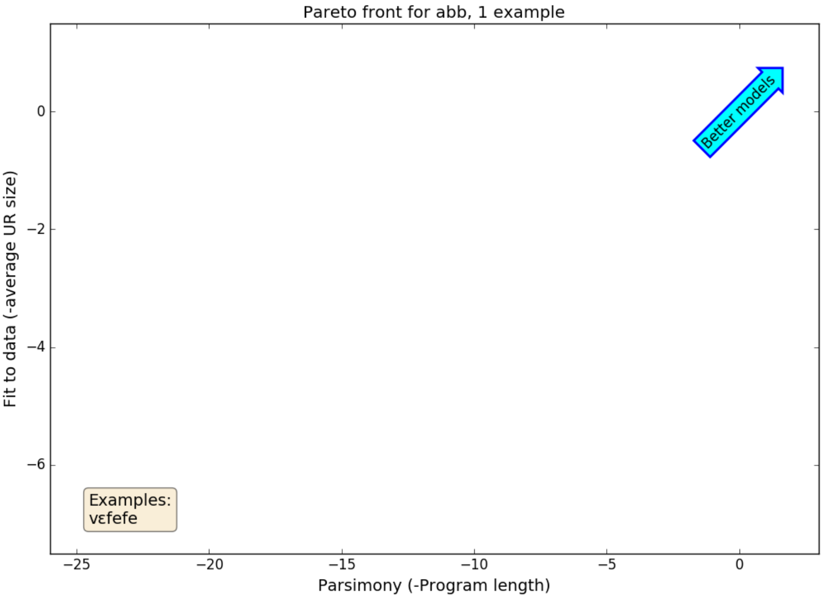
Marcus 1999: Babies learn these grammars

Gerken 2006: Babies learn these grammars *from only a few examples*

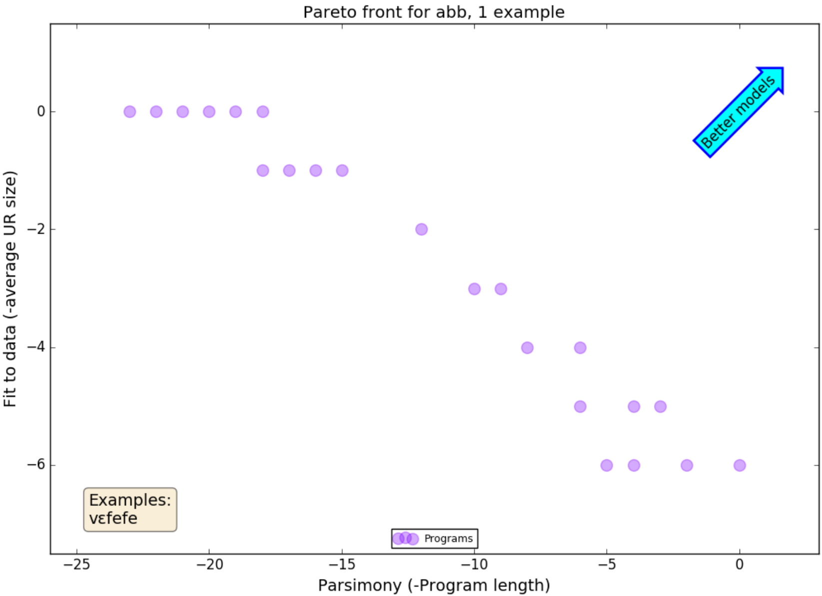
Optimal simplicity front for learning ABB from 1 example



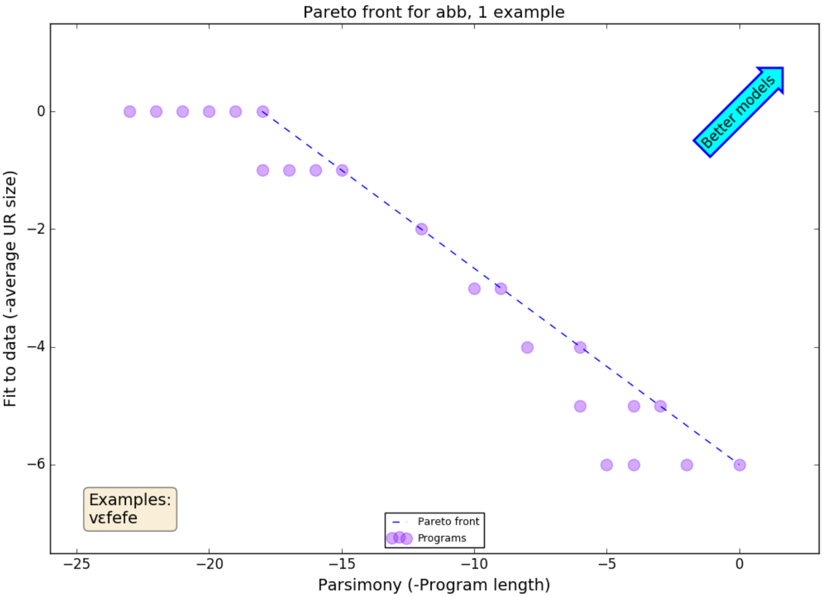
Optimal simplicity front for learning ABB from 1 example



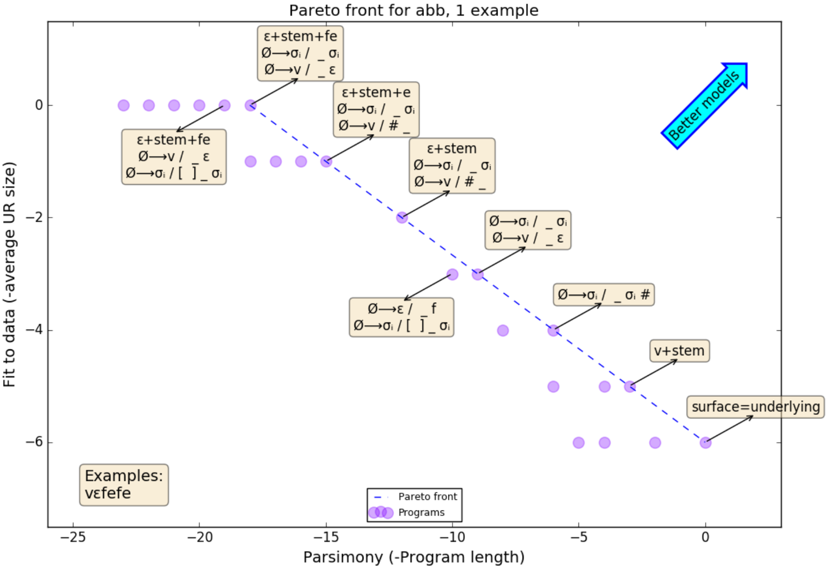
Optimal simplicity front for learning ABB from 1 example



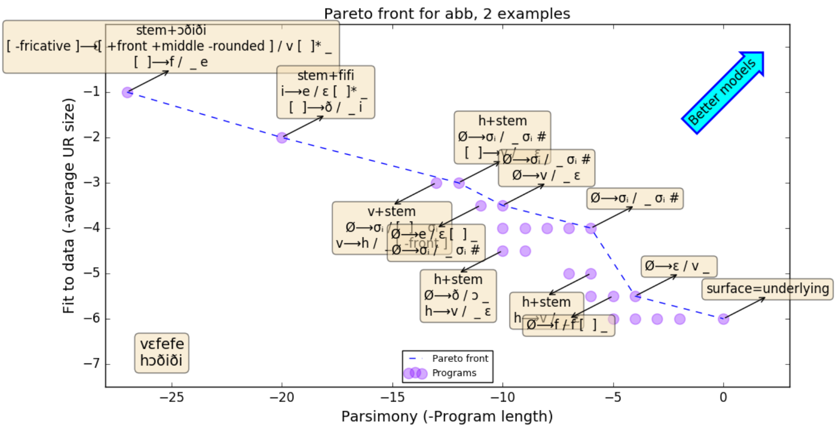
Optimal simplicity front for learning ABB from 1 example



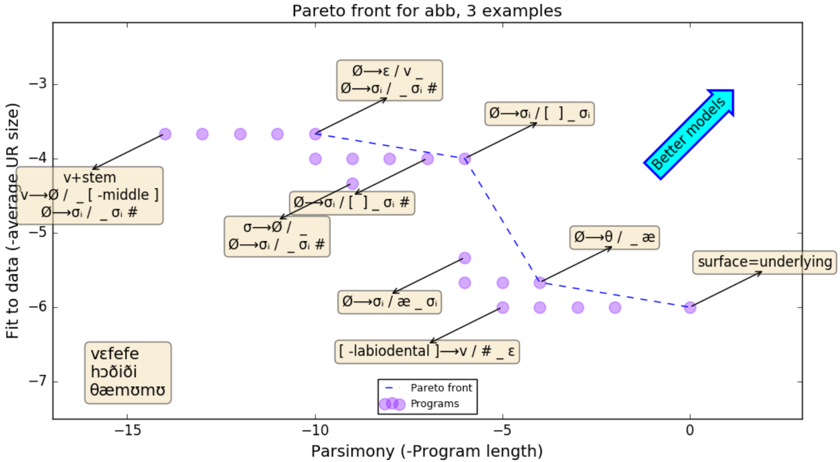
Optimal simplicity front for learning ABB from 1 example



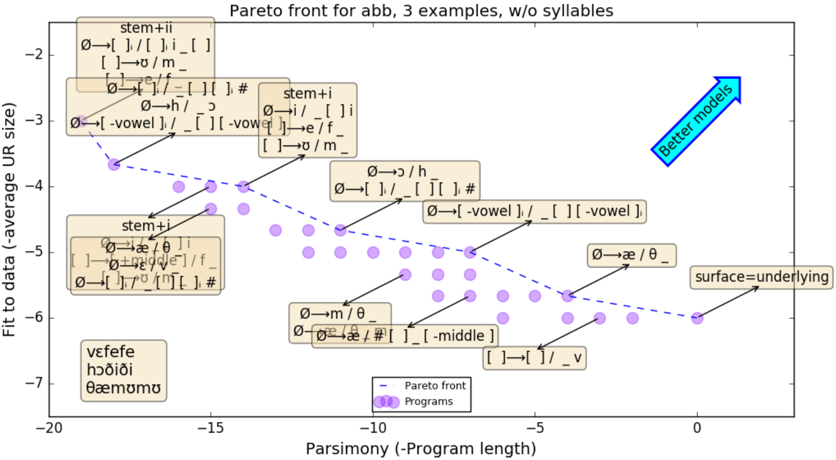
Optimal simplicity front for learning ABB: 2 examples



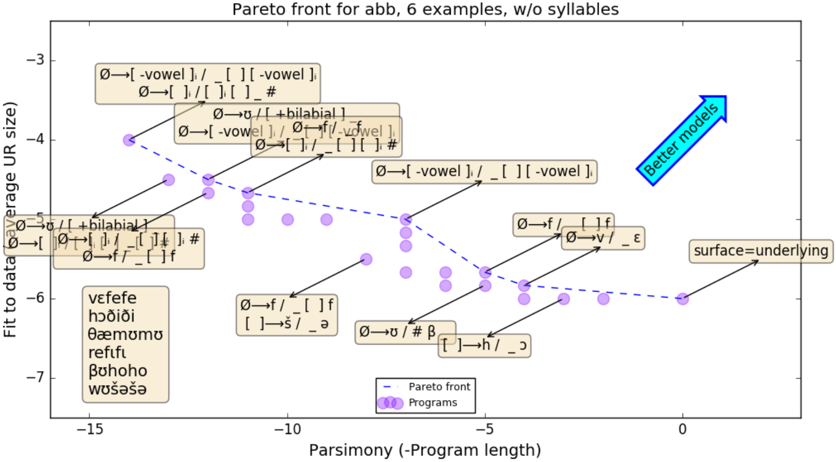
Optimal simplicity front for learning ABB: 3 examples



Optimal simplicity front for learning ABB without syllables, 3 examples



Optimal simplicity front for learning ABB without syllables, 6 examples



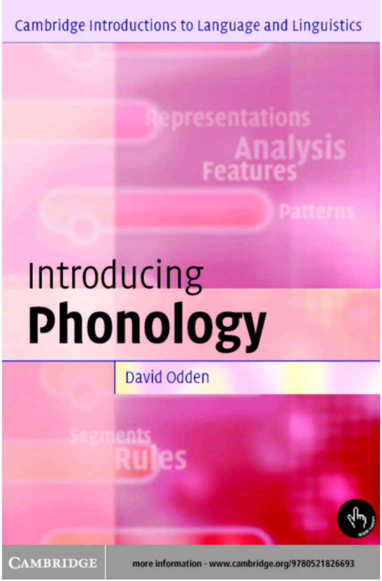
Learning natural language phonology

The vision: Induce programs describing diverse phonological systems from many languages. Imagine getting data from field linguists

Ours is not the first linguistic rule learner: Ezer Rasin & Roni Katzir (this workshop); Albright & Hayes 2003; Yip & Sussman 1996; Constantine Lignos & Charles Yang 2010; Doyle & Bicknell & Levy 2014; Colin Wilson 2006

What is a good stepping stone toward this vision?

Doing your phonology homework



Doing your phonology homework

Exercises

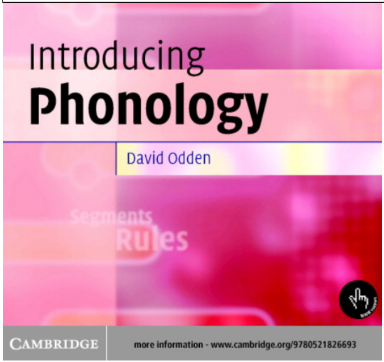
1 Axininca Campa

Provide underlying representations and a phonological rule which will account for the following alternations:

toniro	'palm'	notoniroti	'my palm'
yaarato	'black bee'	noyaaratoti	'my black bee'
kanari	'wild turkey'	noyanariti	'my wild turkey'
kosiri	'white monkey'	noyosiriti	'my white monkey'
pisiro	'small toucan'	nowisiroti	'my small toucan'
porita	'small hen'	noworitati	'my small hen'

Doing your phonology homework

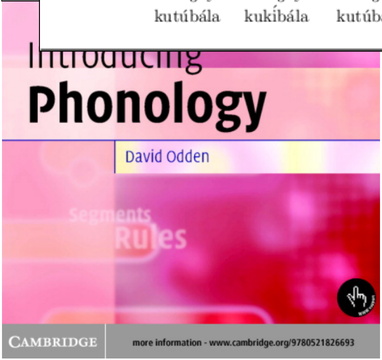
Language	Example data	Phonology	Morphology	Phenomena
Makonde	amáŋga amíle áma akáŋga akíle áka aváŋga avíle óva amáŋga amíle óma utáŋga utíle úta aváŋga avíle éva taváŋga tavíle táva uŋgáŋga uŋgíle úŋga patáŋga patíle póta	[]→[-stress] / _ [ ]* [+stress] [+middle -stress]→a/ _ [ ]	stem+áŋga stem+íle stem+a	Stress patterns Neutralizing rule



Doing your phonology homework

Language	Example data			Phonology	Morphology	Phenomena
	amáŋga	amíle	áma			
	akáŋga	akíle	áka			
Ma						

Language	Example data				Phonology	Morphology
						ku+stem+a
	kubala	kubalana	kubalila	kubalilana		ku+stem+ana
	kugaya	kugayana	kugayila	kugayilana		ku+stem+ila
Kerewe	kubála	kubálána	kubálíla	kubálílana	[] → [-H] / [+H] [] _	ku+stem+ilana
	kutúbála	kukíbála	kutúbálila	kukítúbalila	[] → [+H] / [+H] [] _ []	kutú+stem+a
	kutúgáya	kukígáya	kutúgáyila	kukítúgayila		kukí+stem+a
	kutúbála	kukíbála	kutúbálila	kukítúbalila		kutú+stem+ila
						kukítú+stem+ila



Doing your phonology homework

Language	Example data			Phonology	Morphology	Phenomena
	amáŋga	amíle	áma			
	akáŋga	akíle	áka			

Language	Example data				Phonology	Morphology
Ma	kubala	kubalana	kubalila	kubalilana		ku+stem+a ku+stem+ana

Language	Example data				Phonology	Morphology	Phenomena

Hungarian	a:g^y
	ö:r
	ku:t
	re:s
	rab
	vi:z
	fal
	test
		a:g^yban
		ö:rben
		ku:dban
		re:zben
		rabban
		vi:zben
		falban
		tezdben
			a:$k^y$to:l
			ö:rtö:l
			ku:tto:l
			re:stö:l
			rapto:l
			vi:stö:l
			falto:l
			testtö:l
				a:g^ynak
				ö:rnek
				ku:tnak
				re:snek
				rabnak
				vi:znek
				falnak
				testnek
					`V→[+mid +tense +front]/`
					`[ +front ] [ ]* _`
					`[ ]→[+voice]/`
					`_ [+bilabial]`
					`[-sonorant]→[-voice]/`
					`_ [-voice]`
						stem
						stem+ban
						stem+to:l
						stem+nak
							Vowel harmony
							Voicing assimilation
							Rule ordering

What it can do right now

- **Kikurai**: fricatives alternate with stops after nasals
- **Farsi**: Trills alternate with flaps
- **Amharic**: alternation between ə & ɛ
- **Gen**: l/r alternation
- **Greek**: Velar stops alternate with palletized versions before front vowels
- **Osage**: coronal stops become dentals before central vowels
- **Kishambaa**: voiced/unvoiced nasals alternate
- **Thai**: stops are unreleased word finally
- **Palauan**: a word initial neutralizing rule
- **Quechua**: Velar becomes uvular when followed by uvular (spreading)
- **Lhasa Tibetan**: no contrast between velar/uvular, or voiced/voiceless stops or fricatives
- **Axininca Campa**: stops become glides
- **Kikuyu**: infinitive prefix can surface as either k or ɣ
- **Korean**: vowel harmony, aspiration only surfaces in certain contexts
- **Hungarian**: vowel harmony and voicing assimilation
- **Kikuria**: vowel harmony
- **Farsi**: a deletion rule explains singular/plural
- **Tibetan**: initial consonant cluster reduction explains counting system
- **Makonde**: stress patterns; unstressed vowels are neutralized
- **North Saami**: 3 neutralization rules explain nominative sg essive
- **Samoan**: 2 deletion rules that explain words that sound the same in one inflection being different in another
- **Russian**: devoicing of word final obstruent
- **English**: verbal inflections (voicing assimilation, epenthesis)
- **Finnish**: nominative/partive explained by vowel raising and vowel harmony
- **Kerewe**: Interacting tone rules
- **Polish**: vowel alternations interacting with devoicing of word final obstruent in singular/plural
- **Ancient Greek**: voicing assimilation, deaspiration, deletion rule. Order matters.
- **Serbo-Croatian**: predictable stress, devoicing, neutralizing, epenthesis
- **Artificial grammar learning**: Pig Latin, ABB, ABA, AxA, ABx, AAx, Chinese duplication

What it can do right now

- **Kikurai**: fricatives alternate with stops after nasals
- **Farsi**: Trills alternate with flaps
- **Amharic**: alternation between ə & ɛ
- **Gen**: l/r alternation
- **Greek**: Velar stops alternate with palletized versions before front vowels
- **Osage**: coronal stops become dentals before central vowels
- **Kishambaa**: voiced/unvoiced nasals alternate
- **Thai**: stops are unreleased word finally
- **Palauan**: a word initial neutralizing rule
- **Quechua**: Velar becomes uvular when followed by uvular (spreading)
- **Lhasa Tibetan**: no contrast between velar/uvular, or voiced/voiceless stops or fricatives
- **Axininca Campa**: stops become glides
- **Kikuyu**: infinitive prefix can surface as either k or ɣ
- **Korean**: vowel harmony, aspiration only surfaces in certain contexts
- **Hungarian**: vowel harmony and voicing assimilation
- **Kikuria**: vowel harmony
- **Farsi**: a deletion rule explains singular/plural
- **Tibetan**: initial consonant cluster reduction explains counting system
- **Makonde**: stress patterns; unstressed vowels are neutralized
- **North Saami**: 3 neutralization rules explain nominative sg essive
- **Samoan**: 2 deletion rules that explain words that sound the same in one inflection being different in another
- **Russian**: devoicing of word final obstruent
- **English**: verbal inflections (voicing assimilation, epenthesis)
- **Finnish**: nominative/partive explained by vowel raising and vowel harmony
- **Kerewe**: Interacting tone rules
- **Polish**: vowel alternations interacting with devoicing of word final obstruent in singular/plural
- **Ancient Greek**: voicing assimilation, deaspiration, deletion rule. Order matters.
- **Serbo-Croatian**: predictable stress, devoicing, neutralizing, epenthesis
- **Artificial grammar learning**: Pig Latin, ABB, ABA, AxA, ABx, AAx, Chinese duplication

Handles 60% of the textbook

Polish: Final devoicing + Rule ordering

singular	plural
dom	domi
kot	koti
lut	lodi
vus	vozi
wuk	wugi
ruk	rogi
bur	bori
vsum	vsumi

Polish: Final devoicing + Rule ordering

singular	plural
dom	domi
kot	koti
lut	lodi
vus	vozi
wuk	wugi
ruk	rogi
bur	bori
vsum	vsumi

o→u/ _ [-nasal +voice]#
[-sonorant]→[-voice] / _ #

---

singular: stem
plural: stem+/i/

TOOLKIT: Scaling program synthesis to learn large grammars

Exercises

1 Serbo-Croatian

These data from Serbo-Croatian have been simplified in two ways, to make the problem more manageable. Vowel length is omitted, and some stresses are omitted. The language has both underlying stresses whose position cannot be predicted – these are not marked in the transcriptions – and a predictable "mobile" stress which is assigned by rule – these are the stresses indicated here. Your analysis should account for how stress is assigned in those words marked with a rule-governed stress: you should not try to write a rule that predicts *whether* a word has a stress assigned by rule versus an underlying stress. Ignore the stress of words with no stress mark (other parts of the phonology of such words must be accounted for). Past-tense verbs all have the same general past-tense suffix, and the difference between masculine, feminine and neuter past-tense involves the same suffixes as are used to mark gender in adjectives.

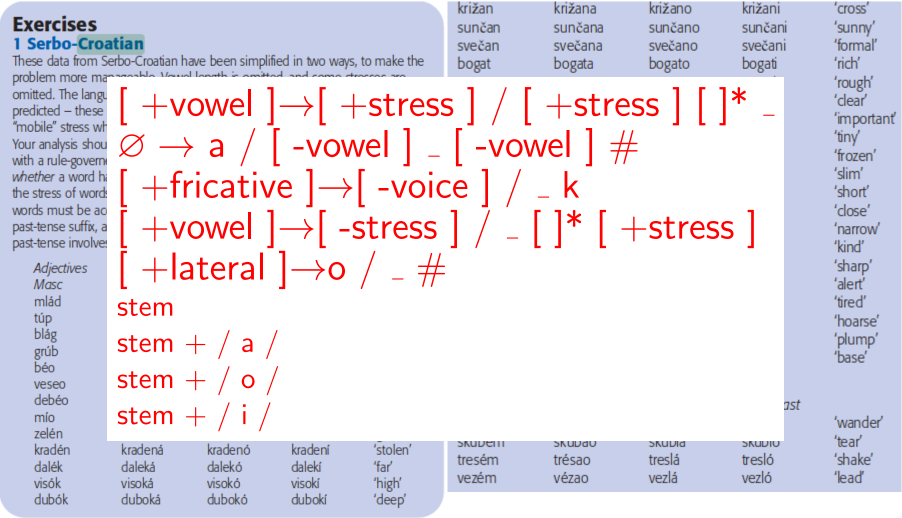
Adjectives

Masc	*Fem*	*Neut*	*Pl*	
mlád	mladá	mladó	mladí	'young'
túp	tupá	tupó	tupí	'blunt'
blág	blagá	blagó	blagí	'mild'
grúb	grubá	grubó	grubí	'coarse'
béo	belá	beló	belí	'white'
veseo	vesela	veselo	veseli	'gay'
debéo	debelá	debeló	debelí	'fat'
mío	milá	miló	milí	'dear'
zelén	zelená	zelenó	zelení	'green'
kradén	kradená	kradenó	kradení	'stolen'
dalék	daleká	dalekó	dalekí	'far'
visók	visoká	visokó	visokí	'high'
dubók	duboká	dubokó	dubokí	'deep'
križan	križana	križano	križani	'cross'
sunčan	sunčana	sunčano	sunčani	'sunny'
svečan	svečana	svečano	svečani	'formal'
bogat	bogata	bogato	bogati	'rich'
rapav	rapava	rapavo	rapavi	'rough'
yásan	yasná	yasnó	yasní	'clear'
vážan	važná	važnó	važní	'important'
sítan	sitná	sitnó	sitní	'tiny'
ledan	ledna	ledno	ledni	'frozen'
tának	tanká	tankó	tankí	'slim'
krátak	kratká	kratkó	kratkí	'short'
blízak	bliská	bliskó	bliskí	'close'
úzak	uská	uskó	uskí	'narrow'
dóbar	dobrá	dobró	dobrí	'kind'
óštar	oštrá	oštró	oštrí	'sharp'
bodar	bodra	bodro	bodri	'alert'
ustao	ustala	ustalo	ustali	'tired'
múkao	muklá	mukló	muklí	'hoarse'
óbao	oblá	obló	oblí	'plump'
pódao	podlá	podló	podlí	'base'

Verbs

1sg pres	*Masc past*	*Fem past*	*Neut past*	
tepém	tépao	teplá	tepló	'wander'
skubém	skúbao	skublá	skubló	'tear'
tresém	trésao	treslá	tresló	'shake'
vezém	vézao	vezlá	vezló	'lead'

TOOLKIT: Scaling program synthesis to learn large grammars



[+vowel]→[+stress] / [+stress] [ ]* _

∅ → a / [-vowel] _ [-vowel] #

[+fricative]→[-voice] / _ k

[+vowel]→[-stress] / _ [ ]* [+stress]

[+lateral]→o / _ #

stem

stem + / a /

stem + / o /

stem + / i /

Investigating the inductive bias over grammars

Motivation: Counting in Tibetan

7 Tibetan

Numbers between 11 and 19 are formed by placing the appropriate digit after the number 10, and multiples of 10 are formed by placing the appropriate multiplier before the number 10. What are the underlying forms of the basic numerals, and what phonological rule is involved in accounting for these data?

ǰu	'10'	ǰig	'1'	ǰugǰig	'11'
ši	'4'	ǰubši	'14'	šibǰu	'40'
gu	'9'	ǰurgu	'19'	gubǰu	'90'
ŋa	'5'	ǰuŋa	'15'	ŋabǰu	'50'

Motivation: Counting in Tibetan

Number	Pronunciation
1	ǰig
4	ši
10	ǰu
11	ǰu+gǰig (10+1)
40	ši+bǰu (4+10)

Motivation: Counting in Tibetan

Number	Pronunciation
1	ǰig
4	ši
10	ǰu
11	ǰu+gǰig (10+1)
40	ši+bǰu (4+10)

Explanation:
ǰig (“1”) is really gǰig
ǰu (“10”) is really bǰu
Program:
$C \rightarrow \varnothing / \# \_ C$
(Delete initial cluster of consonants)

Counting in Tibetan

Number	Pronunciation
1	ǰig
4	ši
10	ǰu
11	ǰu+gǰig (10+1)
40	ši+bǰu (4+10)

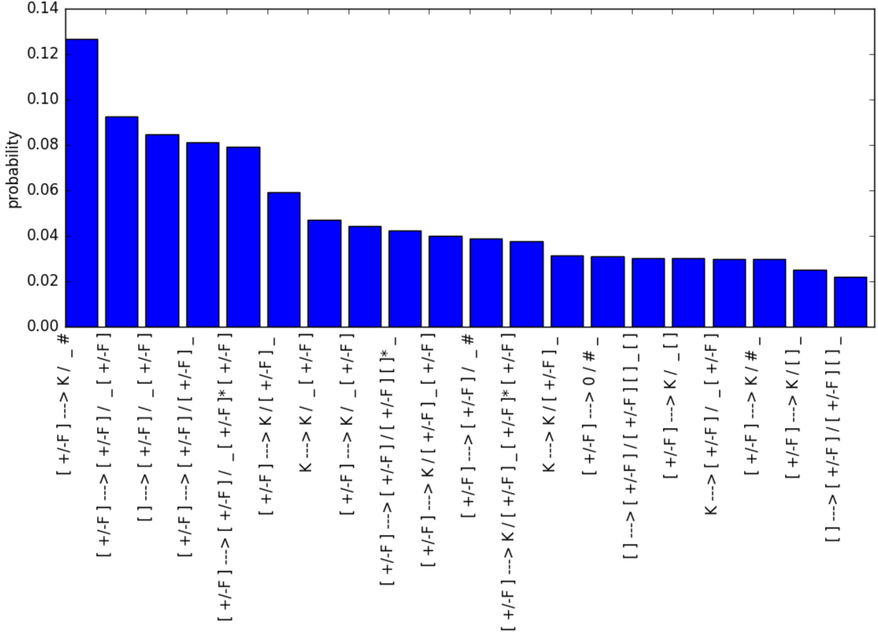
Explanation:
ǰig ("1") is really gǰig
ǰu ("10") is really bǰu
Program:
[-nasal]$\rightarrow \varnothing / \# \_$
Delete initial nonnasal???

Problem:
Probable grammar $\neq$ grammars children/linguists prefer

Problem:
Probable grammar $\neq$ grammars children/linguists prefer

Solution:
Get a better inductive bias (a better universal grammar)

Learning what rules look like



Learning a (fragment) grammar over grammars

Fragment grammars: O'Donnell 2011

FEATUREMATRIX →	[-voice]
FEATUREMATRIX →	[-sonorant]
FEATUREMATRIX →	[+voice]
FEATUREMATRIX →	[+tense]
FEATUREMATRIX →	[+highTone]
FEATUREMATRIX →	[+middle]
TRIGGER →	[+sibilant]
TRIGGER →	[ ] [+highTone]
TRIGGER →	[+stop +voice]
TRIGGER →	[-low]
TRIGGER →	[ ]* FEATUREMATRIX
RULE →	[-vowel] → ∅ / TRIGGER _ TRIGGER
RULE →	[+low] → CONSTANT / [ ]* FEATUREMATRIX _
RULE →	[ ] → [+voice] / _ TRIGGER
RULE →	CONSTANT → CONSTANT / [ ] _ TRIGGER
RULE →	[ ] → [-highTone] / TRIGGER _ TRIGGER
RULE →	[-sonorant] → [-voice] / _ #
RULE →	[ ] → FC / TRIGGER _ [ ] FEATUREMATRIX

Learning a (fragment) grammar over grammars

Fragment grammars: O'Donnell 2011

FEATUREMATRIX →	[-voice]
FEATUREMATRIX →	[-sonorant]
FEATUREMATRIX →	[+voice]
FEATUREMATRIX →	[+tense]
FEATUREMATRIX →	[+highTone]
FEATUREMATRIX →	[+middle]
TRIGGER →	[+sibilant]
TRIGGER →	[ ] [+highTone]
TRIGGER →	[+stop +voice]
TRIGGER →	[-low]
TRIGGER →	[ ]* FEATUREMATRIX
RULE →	[-vowel] → ∅ / TRIGGER _ TRIGGER
RULE →	[+low] → CONSTANT / [ ]* FEATUREMATRIX _
RULE →	[ ] → [+voice] / _ TRIGGER
RULE →	CONSTANT → CONSTANT / [ ] _ TRIGGER
RULE →	[ ] → [-highTone] / TRIGGER _ TRIGGER
RULE →	[-sonorant] → [-voice] / _ #
RULE →	[ ] → FC / TRIGGER _ [ ] FEATUREMATRIX

Learning a (fragment) grammar over grammars

Fragment grammars: O'Donnell 2011

FEATUREMATRIX → [-voice]
FEATUREMATRIX → [-sonorant]
FEATUREM
FEATUREM
FEATUREM
FEATUREM
TRIGGER –
TRIGGER –

In light of this UG, the system revises its rule for Tibetan to match the good linguistics students judgment:
[-vowel] → ∅ / # _ [-vowel]

TRIGGER → [+stop +voice]
TRIGGER → [-low]
TRIGGER → [ ]* FEATUREMATRIX
RULE → [-vowel] → ∅ / TRIGGER _ TRIGGER
RULE → [+low] → CONSTANT / [ ]* FEATUREMATRIX _
RULE → [ ] → [+voice] / _ TRIGGER
RULE → CONSTANT → CONSTANT / [ ] _ TRIGGER
RULE → [ ] → [-highTone] / TRIGGER _ TRIGGER
RULE → [-sonorant] → [-voice] / _ #
RULE → [ ] → FC / TRIGGER _ [ ] FEATUREMATRIX

Problems with minimum description length

Problems with compression

varit	varihin
oahpis	oahpisin
bissobeahtset	bissobeahtsehin
yaaʔmin	yaaʔmimin
gahpir	gahpirin
gaauhtsis	gaauhtsisin
beštor	beštorin
heevemeahhtun	heevemeahhtunin
bissomeahtun	bissomeahtumin
laðas	laðasin
heaŋkkan	heaŋkkanin
yaman	yamanin

Problems with compression

varit	varihin
oahpis	oahpisin
bissobeahtset	bissobeahtsehin
yaaʔmin	yaaʔmimin
gahpir	gahpirin
gaauhtsis	gaauhtsisin
beštor	beštorin
heevemeahhtun	heevemeahhtunin
bissomeahtun	bissomeahtumin
laðas	laðasin
heaŋkkan	heaŋkkanin
yaman	yamanin

stem+/in/

m→n word finally

yaaʔmim→yaaʔmin

yaaʔmim+in→yaaʔmimin

Problems with compression

varit	varihin
oahpis	oahpisin
bissobeahtset	bissobeahtsehin
yaaʔmin	yaaʔmimin
gahpir	gahpirin
gaauhtsis	gaauhtsisin
beštor	beštorin
heevemeahhtun	heevemeahhtunin
bissomeahtun	bissomeahtumin
laðas	laðasin
heaŋkkan	heaŋkkanin
yaman	yamanin

`stem`+/in/

m→n word finally

yaaʔmim→yaaʔmin

yaaʔmim+in→yaaʔmimin

Problems with compression

varit	varihin
oahpis	oahpisin
bissobeahtset	bissobeahtsehin
yaaʔmin	yaaʔmimin
gahpir	gahpirin
gaauhtsis	gaauhtsisin
beštor	beštorin
heevemeahhtun	heevemeahhtunin
bissomeahtun	bissomeahtumin
laðas	laðasin
heaŋkkan	heaŋkkanin
yaman	yamanin

stem+/im/

m→n word finally

yaaʔmim→yaaʔmin

yaaʔmim+im→

yaaʔmimim→

yaaʔmimin

Problems with compression

varit	varihin
oahpis	oahpisin
bissobeahtset	bissobeahtsehin
yaaʔmin	yaaʔmimin
gahpir	gahpirin
gaauhtsis	gaauhtsisin
beštor	beštorin
heevemeahhtun	heevemeahhtunin
bissomeahtun	bissomeahtumin
laðas	laðasin
heaŋkkan	heaŋkkanin
yaman	yamanin

stem+/im/

m→n word finally

yaaʔmim→yaaʔmin

yaaʔmim+im→

yaaʔmimim→

yaaʔmimin

MDL unchanged!

Problems with compression

varit	varihin
oahpis	oahpisin
bissobeahtset	bissobeahtsehin
yaaʔmin	yaaʔmimin
gahpir	gahpirin
gaauhtsis	gaauhtsisin
beštor	beštorin
heevemeahhtun	heevemeahhtunin
bissomeahtun	bissomeahtumin
laðas	laðasin
heaŋkkan	heaŋkkanin
yaman	yamanin

`stem`+/im/

m→n word finally

yaaʔmim→yaaʔmin

yaaʔmim+im→

yaaʔmimim→

yaaʔmimin

MDL

unchanged!

$$\text{cost}(\text{model}; \text{data}) =$$
$$\text{cost}(\text{model}) + \sum_{x \in \text{data}} \text{cost}(x; \text{model}) + \text{runtimeCost}(x; \text{model})$$

Contributions

Engineered new approaches to grammar learning drawn from program synthesis. Allows exact study of simplicity trade-offs between different grammars.

A system that can learn many morphophonologies, both natural and artificial, from relatively small amounts of data

Learning to learn phonological rules as a way of estimating and probing universal grammar

Contributions

Engineered new approaches to grammar learning drawn from program synthesis. Allows exact study of simplicity trade-offs between different grammars.

A system that can learn many morphophonologies, both natural and artificial, from relatively small amounts of data

Learning to learn phonological rules as a way of estimating and probing universal grammar

The end.