Mathematical Linguistics in the 21st Century

Jeffrey Heinz

New Orleans, LA
Workshop on Formal Language Theory
Society for Computation in Language
January 5, 2020

Thesis

Far from being a fossil from a former era, mathematical thinking about language

1. continues to play an essential role in understanding natural languages and
2. continues to make critical contributions to our understanding of how things which compute—both humans and machines—can learn.

Part I

What is mathematics?

What is mathematics?

Marcus Kracht (Los Angeles circa 2005)

"It is a way of thinking."

Eugenia Cheng *How to Bake* π (2015) : 8

"Math, like recipes, has both ingredients and method. . . . In math, the method is probably even more important than the ingredients."

Abstraction

Eugenia Cheng *How to Bake* π (2015) : 16/22

- “Math is there to make things simpler, by finding things that look the same if you ignore some small details.”
- “Abstraction can appear to take you further and further away from reality, but really you’re getting closer and closer to the heart of the matter.”

Abstraction

Eugenia Cheng *How to Bake* π (2015) : 16/22

- "Math is there to make things simpler, by finding things that look the same if you ignore some small details."
- "Abstraction can appear to take you further and further away from reality, but really you're getting closer and closer to the heart of the matter."

Noam Chomsky *The Minimalist Program* (1995) : 6

"*Idealization*, it should be noted, is a misleading term for the only reasonable way to approach a grasp of reality."

I disagree with the word 'only,' but I do think abstraction is underappreciated.

Abstraction

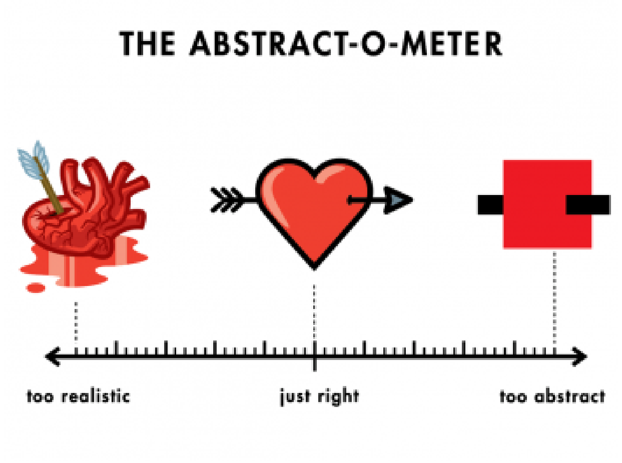
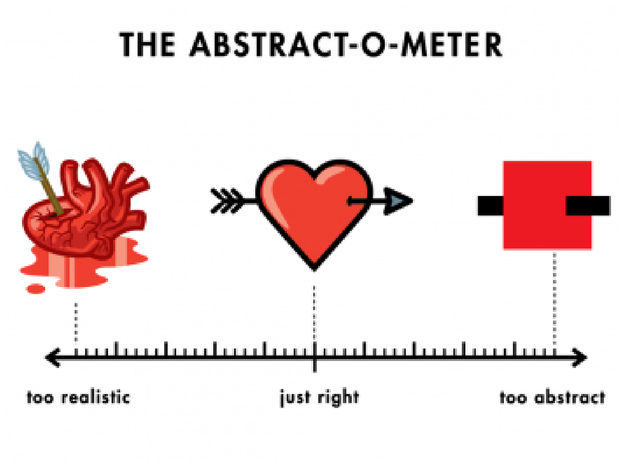


image credit: `https://computersciencewiki.org/index.php/Abstraction`

Abstraction



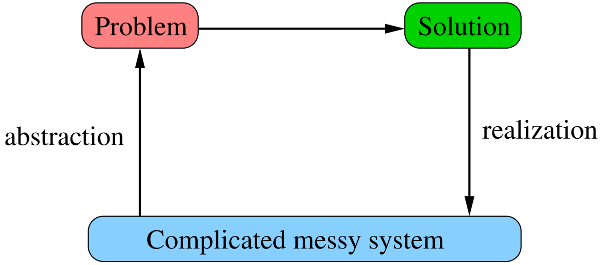
Many things were at one time considered to be “too abstract”:
0, real numbers, $\sqrt{-1}$, uncountable infinity, number theory, …

Goals

Deducing consequences from premises.

Advantages:

1. Can provide complete, verifiable, interpretable & understandable *solutions* to *problems*.
2. Can provide fresh insight into reality.
3. Truth is timeless.

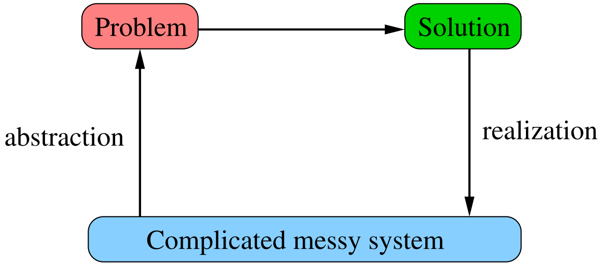


Goals

Deducing consequences from premises.

Disadvantages:

1. The 'abstraction' and 'realization' steps take additional work and time.



Part II

Overview of Rest of Talk

Mathematical Linguistics in the 21st Century

Timeframe

- Of course much is owed to Boole and Frege of the 19th century
- Of course much is owed to Russell, Church, Turing, Post, the Polish school of Logic, and others
- Of course there are many others I omit (Montague, Bach, Mönnich, Savitch, Johnson, …)
- I am going to focus on specific contributions in the latter half of the 20th century which directly contributed to my own interests.

Linguistic Questions as Computational Problems

What is it that we know when we know a language?

How do we come by this knowledge?

Linguistic Questions as Computational Problems

What is it that we know when we know a language?

How do we come by this knowledge?

1. A Membership problem.
2. A Learning problem.
3. Variations thereof.

Part III

Characterizing knowledge of language

Conservativity (Keenan and Stavi 1986)

The theory of Generalized Quantifiers addresses determiner expressions like

- *every, all, some, not one, more than three, fewer than twenty, most, how many, which, more male than female, less than half, . . .*

in utterances like

> ____ birds fly south for the winter.

1. What are the (possible) denotations of these expressions?
2. How arbitrary can they be?

Conservativity (Keenan and Stavi 1986)

All birds fly south for the winter.

Denotation of **all**

- ALL P Q is true iff P$\subseteq$ Q

Conservativity (Keenan and Stavi 1986)

All birds fly south for the winter.

Denotation of **all**

- ALL P Q is true iff P$\subseteq$ Q

Conservativity

D is **conservative** iff D P Q = D P R whenever P$\cap$Q = P$\cap$R.

Informally, this means that in evaluating D P Q we ignore the elements of Q which do not lie in P.

Conservativity (Keenan and Stavi 1986)

All birds fly south for the winter.

Denotation of **all**

- ALL P Q is true iff P$\subseteq$ Q

Conservativity

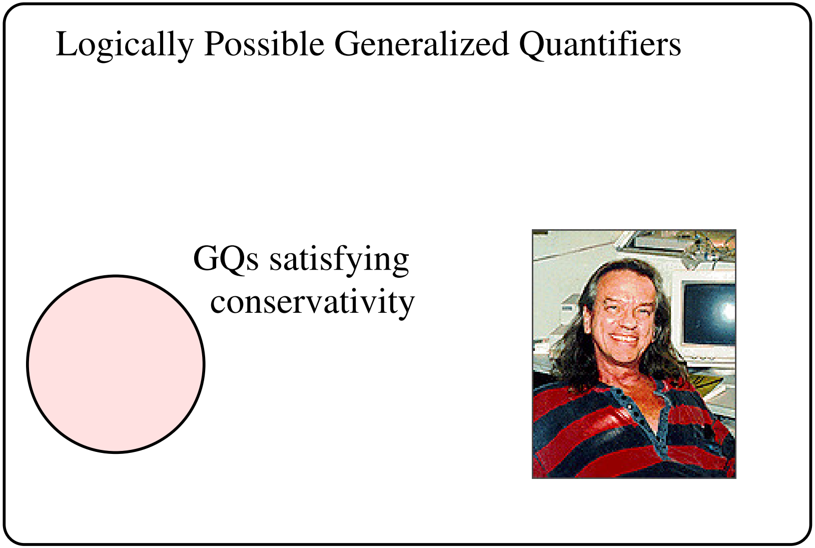
D is **conservative** iff D P Q = D P R whenever P$\cap$Q = P$\cap$R.

Informally, this means that in evaluating D P Q we ignore the elements of Q which do not lie in P.

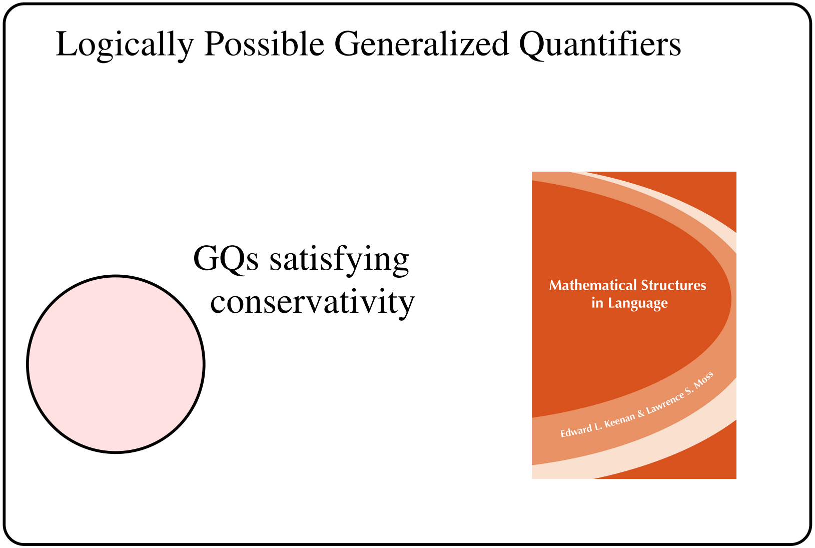
An example of non-conservative D:

- HÄRTIG P Q is true iff $|P| = |Q|$

Conservativity (Keenan and Stavi 1986)


Logically Possible Generalized Quantifiers
GQs satisfying conservativity

Conservativity (Keenan and Stavi 1986)


Logically Possible Generalized Quantifiers
GQs satisfying conservativity

Mathematical Structures in Language

Edward L. Keenan & Lawrence S. Moss

Mathematics of Sequences

- Language unfolds over time.
- We observe sequences of linguistic events.
- What is the mathematics of sequences?
- What is the mathematics of other relational structures like trees and graphs?

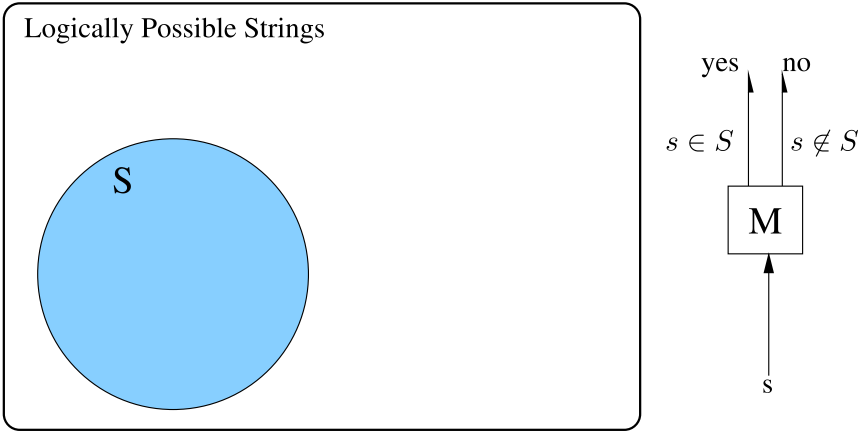
Mathematics of Sequences

- Language unfolds over time.
- We observe sequences of linguistic events.
- What is the mathematics of sequences?
- What is the mathematics of other relational structures like trees and graphs?

Knowledge of language includes knowledge of which sequences are licit and which are not.

- John laughed and laughed. ✓
- John and laughed. ✗

A Membership problem



Variations thereof

Functions on the string domain …

Function Type	Output Type
$\Sigma^* \to \{T, F\}$	Booleans
$\Sigma^* \to \Sigma^*$	Strings
$\Sigma^* \to \mathbb{N}$	Natural Numbers
$\Sigma^* \to [0,1]$	Reals in the Unit Interval
$\Sigma^* \to P(\Sigma^*)$	Stringsets
…	

Variations thereof

Functions on the string domain ...

Function Type	Output Type
$\Sigma^* \to \{T, F\}$	Booleans
$\Sigma^* \to \Sigma^*$	Strings
$\Sigma^* \to \mathbb{N}$	Natural Numbers
$\Sigma^* \to [0,1]$	Reals in the Unit Interval
$\Sigma^* \to P(\Sigma^*)$	Stringsets
...	

Mathematics classifies **numerical** functions according to general properties: linear, polynomial, trigonometric, logarithmic, ...

Variations thereof

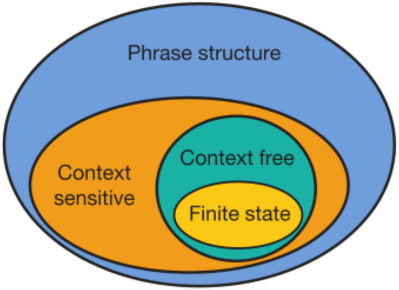
Functions on the string domain …

Function Type	Output Type
$\Sigma^* \rightarrow \{T, F\}$	Booleans
$\Sigma^* \rightarrow \Sigma^*$	Strings
$\Sigma^* \rightarrow \mathbb{N}$	Natural Numbers
$\Sigma^* \rightarrow [0,1]$	Reals in the Unit Interval
$\Sigma^* \rightarrow P(\Sigma^*)$	Stringsets
…	

Mathematics classifies **numerical** functions according to general properties: linear, polynomial, trigonometric, logarithmic, …

How can we classify functions like those above?

Classifying Membership Problems



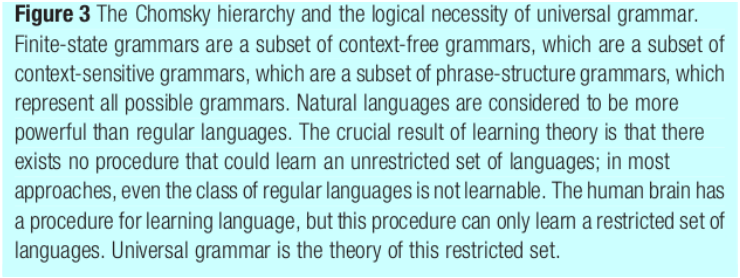


Figure 3 The Chomsky hierarchy and the logical necessity of universal grammar. Finite-state grammars are a subset of context-free grammars, which are a subset of context-sensitive grammars, which are a subset of phrase-structure grammars, which represent all possible grammars. Natural languages are considered to be more powerful than regular languages. The crucial result of learning theory is that there exists no procedure that could learn an unrestricted set of languages; in most approaches, even the class of regular languages is not learnable. The human brain has a procedure for learning language, but this procedure can only learn a restricted set of languages. Universal grammar is the theory of this restricted set.

Nowak et al. 2002, *Nature*

Classifying Membership Problems

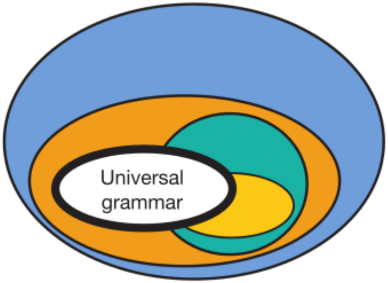
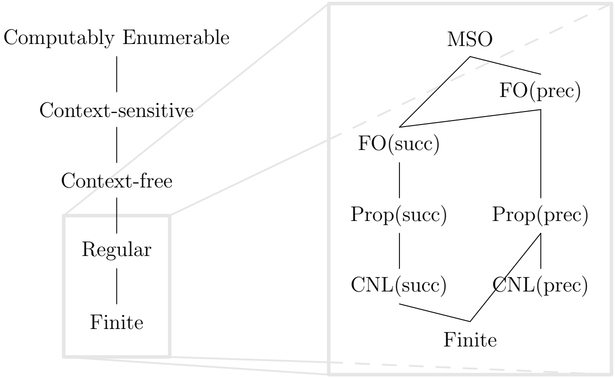


Figure 3 The Chomsky hierarchy and the logical necessity of universal grammar. Finite-state grammars are a subset of context-free grammars, which are a subset of context-sensitive grammars, which are a subset of phrase-structure grammars, which represent all possible grammars. Natural languages are considered to be more powerful than regular languages. The crucial result of learning theory is that there exists no procedure that could learn an unrestricted set of languages; in most approaches, even the class of regular languages is not learnable. The human brain has a procedure for learning language, but this procedure can only learn a restricted set of languages. Universal grammar is the theory of this restricted set.

Nowak et al. 2002, *Nature*

Where is natural language?



1. Morpho-phonology is regular (Johnson 1972, Kaplan and Kay 1994, Roark and Sproat 2007, a.o.)
2. Syntax is mildly context sensitive (Joshi 1984, Schieber 1985, Joshi, Vijay-Shanker & Weir 1991, Stabler 1997, a.o.)

From the 20th to the 21st Century

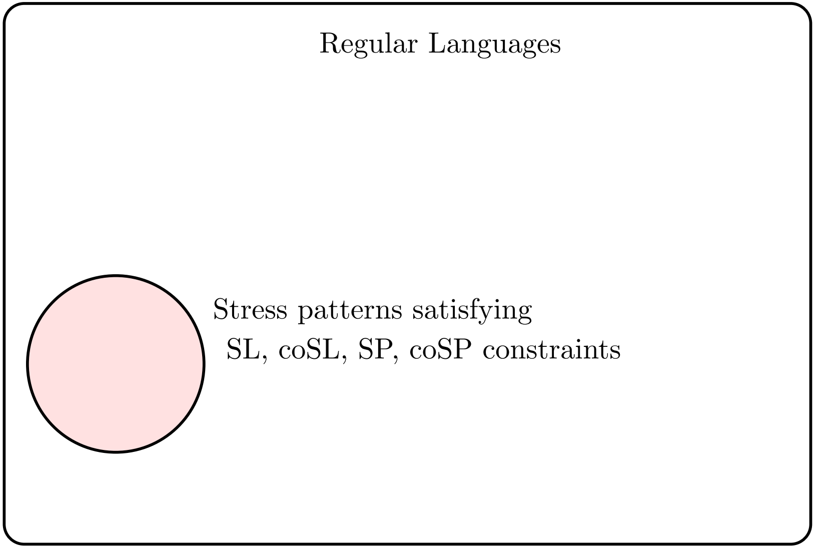
Example #1: Stress Patterns Rogers and Lambert (2019, JLM)

1. They consider over 100 distinct stress patterns, expressed as regular grammars, from over 700 languages in the StressTyp2 database.
2. They develop methods to *factor* these grammars into primitive constraints. Almost all constraints fall into these kinds:
    1. Strictly Local (no LL)
    2. Co-Strictly Local (require Ĺ)
    3. Strictly Piecewise (no Ḱ...L)
    4. Co-Strictly Piecewise (require Ḱ...L)

Example #1: Stress Patterns Rogers and Lambert (2019, JLM)

1. They consider over 100 distinct stress patterns, expressed as regular grammars, from over 700 languages in the StressTyp2 database.
2. They develop methods to *factor* these grammars into primitive constraints. Almost all constraints fall into these kinds:
    1. Strictly Local (no LL)
    2. Co-Strictly Local (require Ĺ)
    3. Strictly Piecewise (no Ĥ...L)
    4. Co-Strictly Piecewise (require Ĥ...L)
3. See also their 2019 MoL paper.

Stress patterns are not just regular, they belong to distinct sub-regular classes.



Example #2: Local and Long-distance string transformations

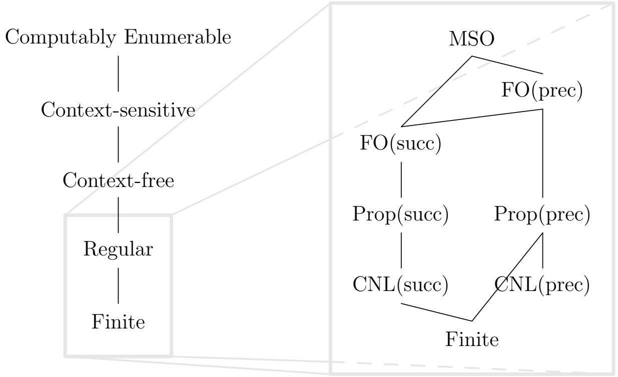
Some facts

1. In phonology, both local and non-local assimilations occur (post-nasal voicing, consonant harmony, . . .)
2. In syntax, both local and non-local dependencies exist (selection, wh-movement, . . .)
3. There is also copying (reduplication). . .

Questions

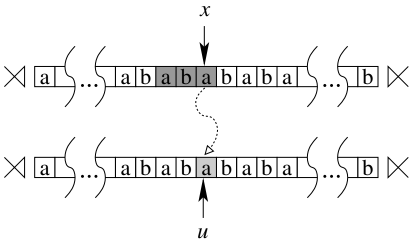
1. What are (possible) phonological processes? Syntactic dependencies?
2. How arbitrary can they be?

What is Local?



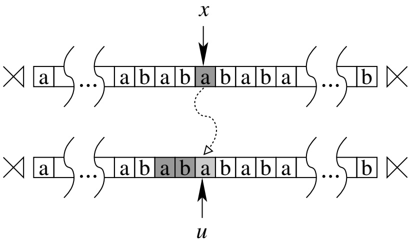
1. The 20th century gave us local and long-distance dependencies in (sets of) sequences
2. But it wasn't until the 21st century that a theory of Markovian/Strictly Local string-to-string functions was developed (Chandlee 2014 et seq.)

Input/Output Strictly Local Functions


x
⋊ a ... a b a b a b a b a ... b ⋉
⋊ a ... a b a b a b a b a ... b ⋉
u

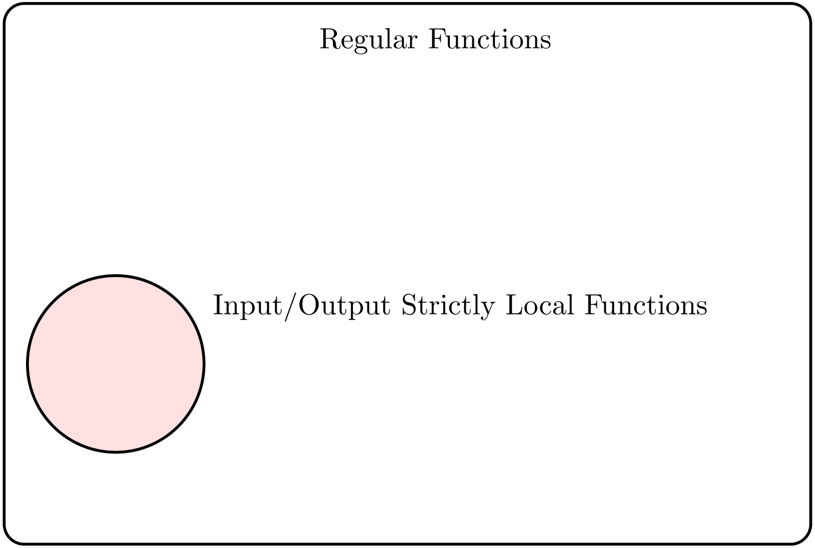
Chandlee 2014 et seq.

Input/Output Strictly Local Functions

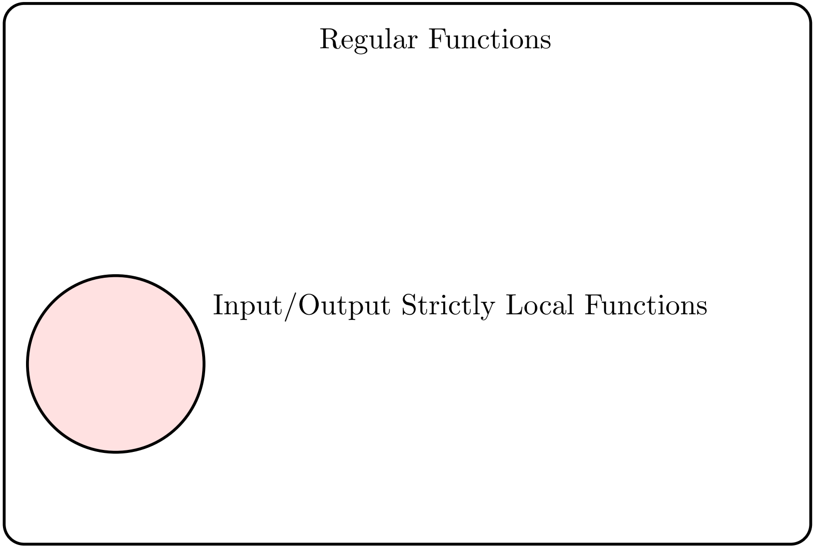


Chandlee 2014 et seq.

How much Phonology is in there?



How much Phonology is in there?



Graf (2020, SCiL) extend this notion of locality to tree functions to characterize subcategorization in syntactic structures

What is Non-Local?

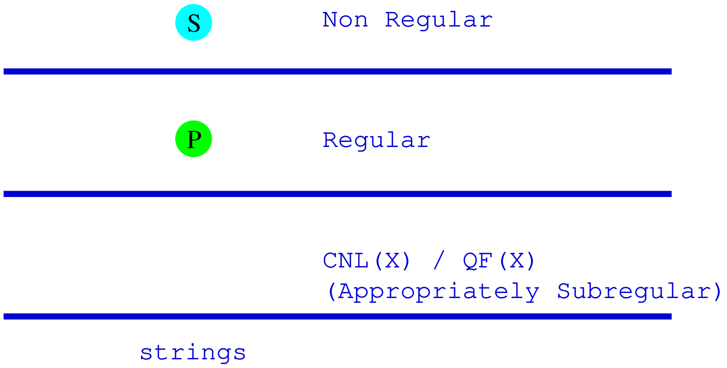
There are **different types** of long-distance dependencies overs strings just like there are different types of non-linear numerical functions.

What is Non-Local?

There are **different types** of long-distance dependencies overs strings just like there are different types of non-linear numerical functions.

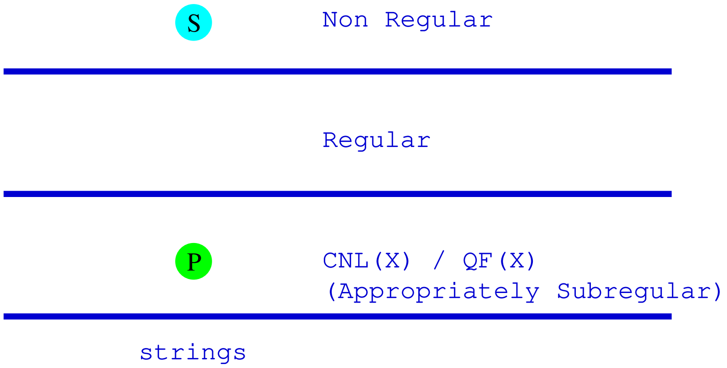
1. Tier-based Strictly Local Functions (McMullin a.o.)
2. Strictly Piecewise Functions (Burness and McMullin, SCiL 2020)
3. Subsequential Functions (Mohri 1997 et seq.)
4. Subclasses of 2way FSTs (Dolatian and Heinz 2018 et seq.)
5. . . .

Example #3: Parallels between Syntax and Phonology



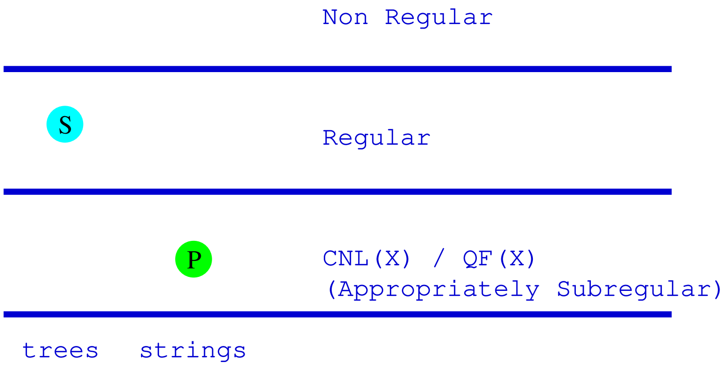
(Chomsky 1957, Johnson 1972, Kaplan and Kay 1994, Roark and Sproat 2007, and many others)

Example #3: Parallels between Syntax and Phonology



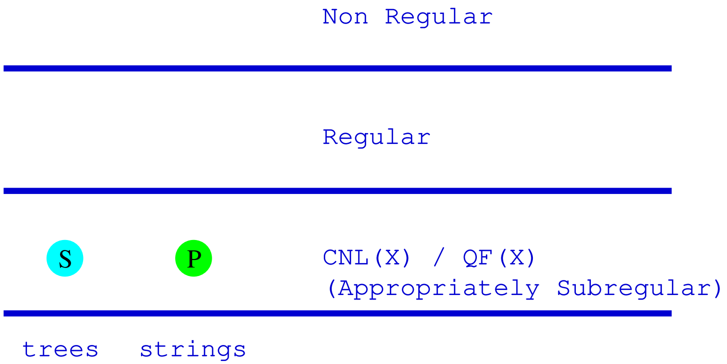
(Potts and Pullum 2002, Heinz 2007 et seq., Graf 2010, Rogers et al. 2010, 2013, Rogers and Lambert, and many others)

Example #3: Parallels between Syntax and Phonology



(Rogers 1994, 1998, Knight and Graehl 2005, Pullum 2007, Kobele 2011, Graf 2011 and many others)

Example #3: Parallels between Syntax and Phonology



(Graf 2013, 2017, Vu et al. 2019, Shafiei and Graf 2020, and others)

Example #4: Other Applications

Understanding Neural Networks:

1. Merril (2019) analyzes the asymptotic behavior of RNNs in terms of regular languages.

Example #4: Other Applications

Understanding Neural Networks:

1. Merril (2019) analyzes the asymptotic behavior of RNNs in terms of regular languages.
2. Rabusseau et al. (2019 AISTATS) proves 2nd-order RNNs are equivalent to weighted finite-state machines.

Example #4: Other Applications

Understanding Neural Networks:

1. Merril (2019) analyzes the asymptotic behavior of RNNs in terms of regular languages.
2. Rabusseau et al. (2019 AISTATS) proves 2nd-order RNNs are equivalent to weighted finite-state machines.
3. Nelson et al. (2020) (SCiL) use Dolatian's analysis of reduplication with 2way finite-state transducers to better understand what and how RNNs with and without attention can learn.

Summary of this Part

What is it that we know when we know a language?

1. Mathematical linguistics in the 20th century, and so far into the 21st century, continues to give us essential insights into (nearly) universal properties of natural languages.

Summary of this Part

What is it that we know when we know a language?

1. Mathematical linguistics in the 20th century, and so far into the 21st century, continues to give us essential insights into (nearly) universal properties of natural languages.
2. This is accomplished by dividing the logically possible space of generalizations into categories and studying where the natural language generalizations occur.

Summary of this Part

What is it that we know when we know a language?

1. Mathematical linguistics in the 20th century, and so far into the 21st century, continues to give us essential insights into (nearly) universal properties of natural languages.
2. This is accomplished by dividing the logically possible space of generalizations into categories and studying where the natural language generalizations occur.
3. The properties are not about a particular formalism (like finite-state vs. regular expressions vs. rules vs. OT) but more about conditions on grammars. What must/should any grammar at least be sensitive to? What can/should be ignored?

Summary of this Part

What is it that we know when we know a language?

1. Mathematical linguistics in the 20th century, and so far into the 21st century, continues to give us essential insights into (nearly) universal properties of natural languages.
2. This is accomplished by dividing the logically possible space of generalizations into categories and studying where the natural language generalizations occur.
3. The properties are not about a particular formalism (like finite-state vs. regular expressions vs. rules vs. OT) but more about conditions on grammars. What must/should any grammar at least be sensitive to? What can/should be ignored?
4. Because it's math, it is verifiable, interpretable, analyzable & understandable, and is thus used to understand complicated systems (like natural languages and NNs).

Part IV

Learning Problems

Questions about learning

Motivating question

How do we come by our knowledge of language?

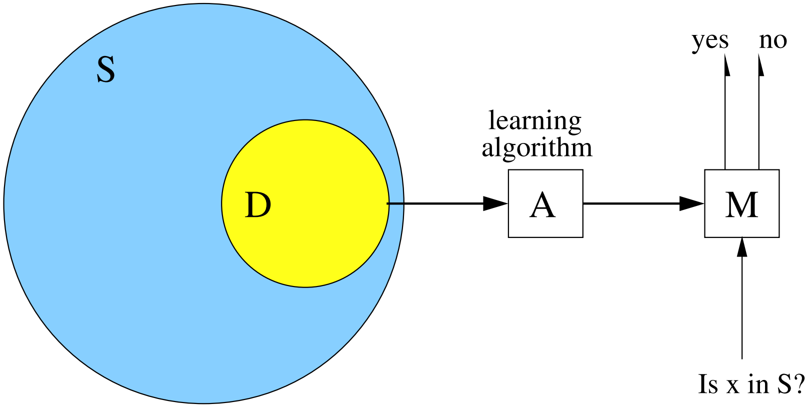
Questions about learning

1. What does it mean to learn?
2. How can learning be a formalized as a problem and solved (like the problem of sorting lists)?

Some answers from before the 21st century: Computational Learning Theory

1. Identifications in the Limit (Gold 1967)
2. Active/Query Learning (Angluin 1988)
3. Probably Approximately Correct (PAC) Learning (Valiant 1984)
4. Optimizing Objective Functions
5. …

In Pictures



for any S belonging to a class C?

Many, many methods

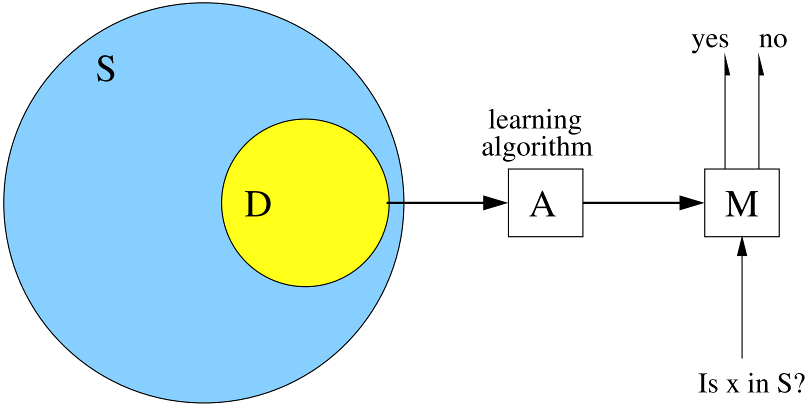
1. Connectionism/Associative Learning (Rosenblatt 1959, McClelland and Rumelhart 1986, Kapatsinski 2018, a.o.)
2. Bayesian methods (Bishop 2006, Kemp and Tenenbaum 2008, a.o.)
3. Probabilistic Graphical Models (Pearl 1988, Koller and Friedman 2010, a.o.)
4. State-merging (Feldman 1972, Angluin 1982, Oncina et al 1992, a.o.)
5. Statistical Relational Learning (De Raedt 2008, a.o.)
6. Minimum Description Length (Risannen 1978, Goldsmith a.o..)
7. Suport Vector Machines (Vapnik 1995, 1998 a.o.)
8. . . .

Many, many methods

Newer methods

1. Deep NNs (LeCun et al. 2015, Schmidhuber 2015, Goodfellow et al. 2016, a. MANY o.)
    - encoder-decoder networks
    - generative adversarial networks
    - ...
2. Spectral Learning (Hsu et al 2009, Balle et al. 2012, 2014, a.o.)
3. Distributional Learning (Clark and Yoshinaka 2016, a.o.)
4. ...

Computational Learning Theory



CLT studies **conditions** on learning mechanisms/methods!

The main lesson from CLT

There is no free lunch.

1. There is **no** algorithm that can feasibly learn **any** pattern P, even with lots of data from P.

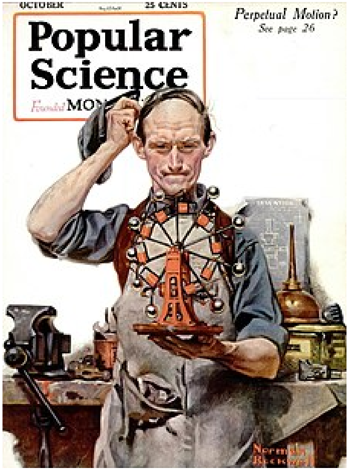
The main lesson from CLT

There is no free lunch.

1. There is **no** algorithm that can feasibly learn **any** pattern P, even with lots of data from P.
2. But—There are algorithms that can feasibly learn patterns **which belong to a suitably structured class C**.

Gold 1967, Angluin 1980, Valiant 1984,
Wolpert and McReady 1997, a.o.

The Perpetual Motion Machine



October 1920 issue of Popular Science magazine, on perpetual motion.

"Although scientists have established them to be impossible under the laws of physics, perpetual motion continues to capture the imagination of inventors."

`https://en.wikipedia.org/wiki/Perpetual_motion`

The Perpetual Misconception Machine

$\exists$ machine-learning algorithm $A, \forall$ patterns P with enough data D from $P : A(D) \approx P$.

1. It's just not true.

The Perpetual Misconception Machine

$\exists$ machine-learning algorithm A, $\forall$ patterns P with enough data D from P : $A(D) \approx P$.

1. It's just not true.
2. What is true is this:
   $\forall$ patterns P, $\exists$ data D and ML A : $A(D) \approx P$.

The Perpetual Misconception Machine

$\exists$ machine-learning algorithm A, $\forall$ patterns P with enough data D from $P : A(D) \approx P$.

1. It's just not true.
2. What is true is this:
   $\forall$ patterns P, $\exists$ data D and ML $A : A(D) \approx P$.
3. In practice, the misconception means searching for A and D so that your approximation is better than everyone else's.

The Perpetual Misconception Machine

$\exists$ machine-learning algorithm A, $\forall$ patterns P with enough data D from P : $A(D) \approx P$.

1. It's just not true.
2. What is true is this:
   $\forall$ patterns P, $\exists$ data D and ML A : $A(D) \approx P$.
3. In practice, the misconception means searching for A and D so that your approximation is better than everyone else's.
4. With next pattern P', we will have no guarantee A will work, we will have to search again.

Computational Laws of Learning

Feasibly solving a learning problem requires defining a target class C of patterns.

1. The class C cannot be all patterns, or even all computable patterns.

Computational Laws of Learning

Feasibly solving a learning problem requires defining a target class C of patterns.

1. The class C cannot be all patterns, or even all computable patterns.
2. Class C must have *more structure*, and many logically possible patterns *must be outside* of C.

Computational Laws of Learning

Feasibly solving a learning problem requires defining a target class C of patterns.

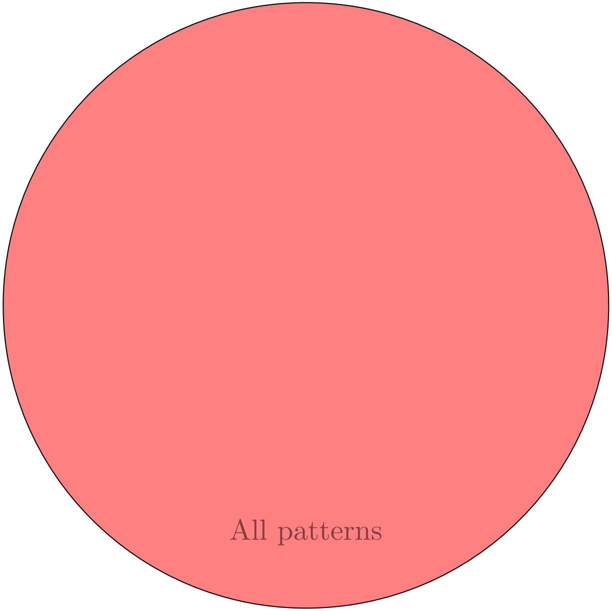
1. The class C cannot be all patterns, or even all computable patterns.
2. Class C must have *more structure*, and many logically possible patterns *must be outside* of C.
3. There is no avoiding prior knowledge.

Computational Laws of Learning

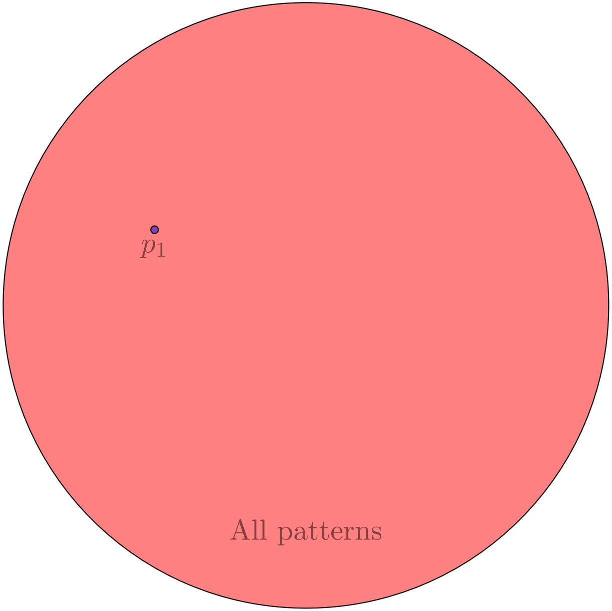
Feasibly solving a learning problem requires defining a target class C of patterns.

1. The class C cannot be all patterns, or even all computable patterns.
2. Class C must have *more structure*, and many logically possible patterns *must be outside* of C.
3. There is no avoiding prior knowledge.
4. Do not "confuse ignorance of biases with absence of biases." (Rawski and Heinz 2019)

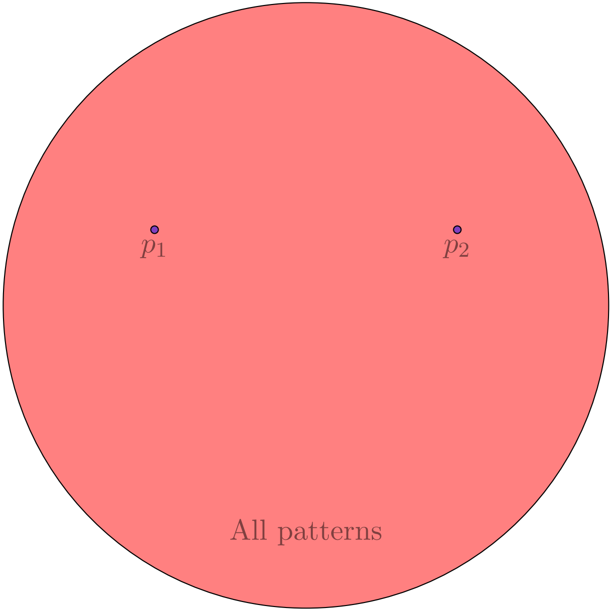
In Pictures: Given ML algorithm A



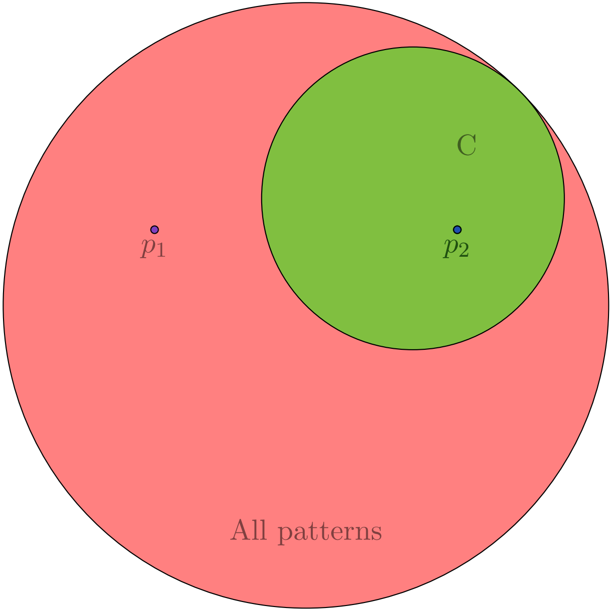
In Pictures: Given ML algorithm A



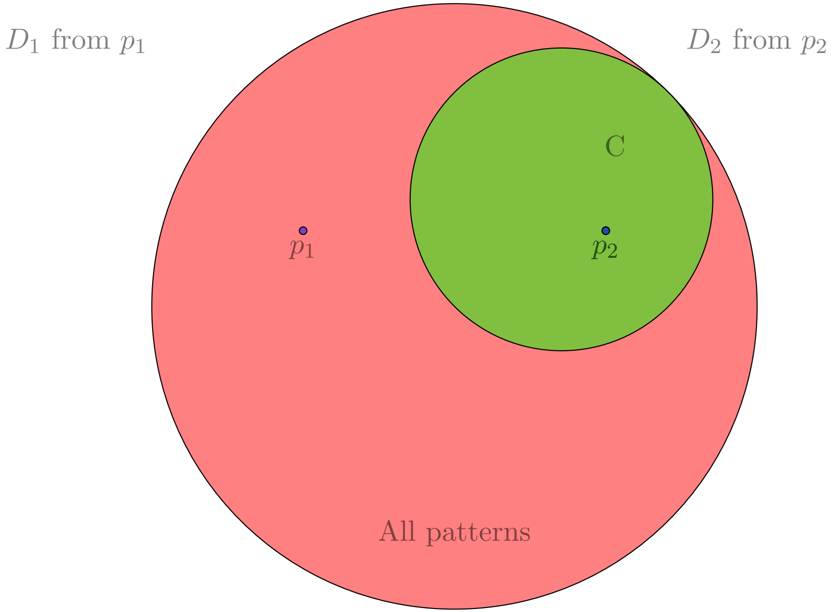
In Pictures: Given ML algorithm A



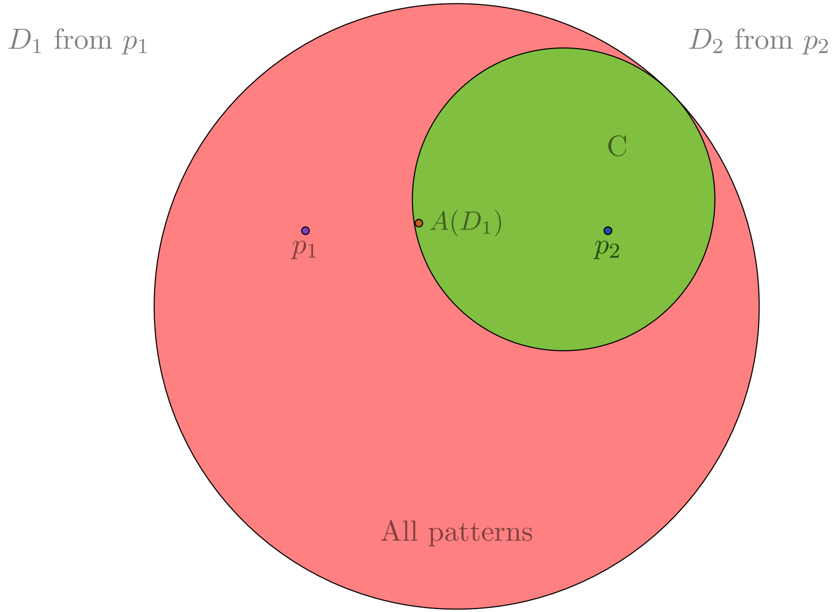
In Pictures: Given ML algorithm A



In Pictures: Given ML algorithm A



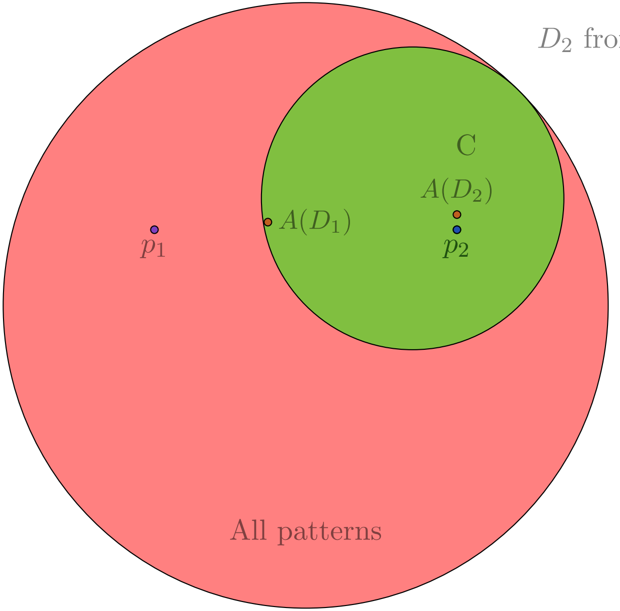
In Pictures: Given ML algorithm A



In Pictures: Given ML algorithm A

D_1 from p_1

D_2 from p_2

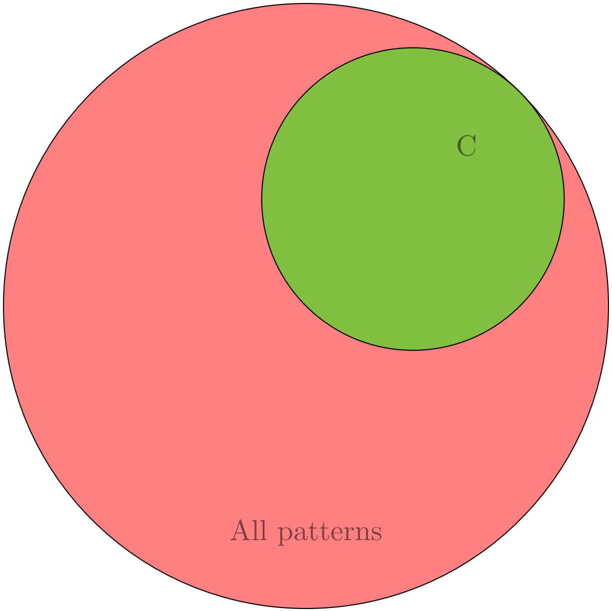


The Perpetual Misconception Machine

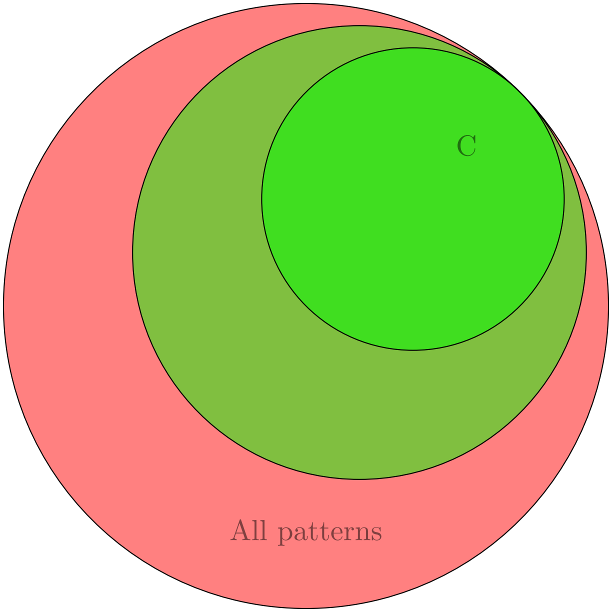
When you believe in things that you don't understand
then you suffer.

– Stevie Wonder

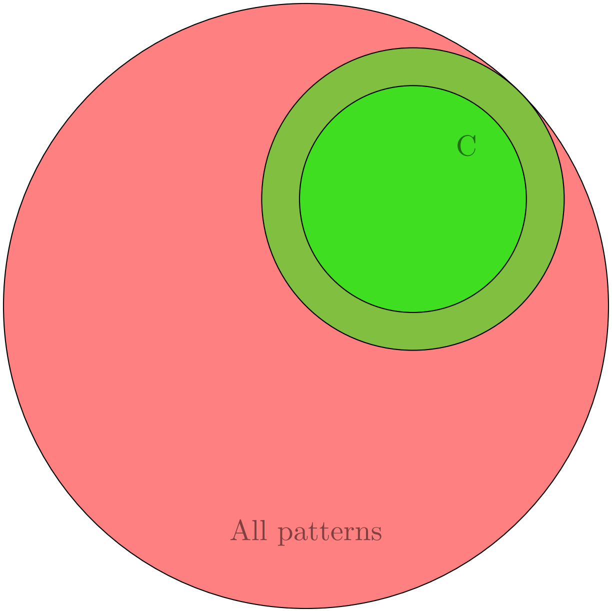
Go smaller, not bigger!



Go smaller, not bigger!



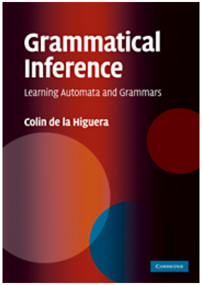
Go smaller, not bigger!

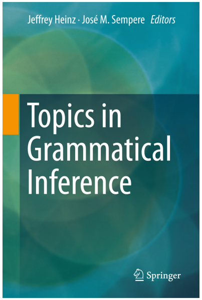


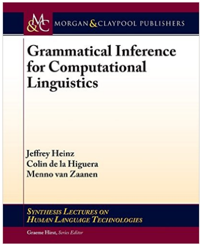
Dana Angluin

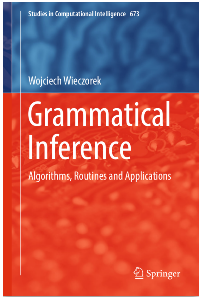
1. Characterized those classes identifiable in the Limit from Positive Data (1980).
2. Introduced first non-trivial infinite class of languages identifiable in the limit from positive data with an efficient algorithm (1982).
3. Introduced Query Learning; Problem and Solution (1987a,b).
4. Studied learning with noise, from stochastic examples (1988a,b).

Grammatical Inference


Grammatical Inference
Learning Automata and Grammars
Colin de la Higuera


Jeffrey Heinz · José M. Sempere *Editors*
Topics in Grammatical Inference
Springer


MORGAN & CLAYPOOL PUBLISHERS
Grammatical Inference for Computational Linguistics
Jeffrey Heinz
Colin de la Higuera
Menno van Zaanen
SYNTHESIS LECTURES ON HUMAN LANGUAGE TECHNOLOGIES
Graeme Hirst, *Series Editor*


Studies in Computational Intelligence 673
Wojciech Wieczorek
Grammatical Inference
Algorithms, Routines and Applications
Springer

ICGI 2020 in NYC August 26-28!!

`https://grammarlearning.org/`

From the 20th to the 21st Century

Example #1: SL, SP, TSL; ISL, OSL, I-TSL, O-TSL

1. These classes are parameterized by a window size k.
2. The k-classes are efficiently learnable from positive examples under multiple paradigms.

(Garcia et al. 1991, Heinz 2007 et seq., Chandlee et al. 2014, 2015, Jardine and McMullin 2017, Burness and McMullin 2019 a.o.)

Example #2: Other Applications

1. Using Grammatical Inference to understand Neural Networks.
    1. Weiss et al. (2018, 2019) use Angluin's L* (1987) algorithm (and more) to model behavior of trained NNs with FSMs.
    2. Eyraud et al. (2018) use spectral learning to model behavior of trained NNs with FSMs.
2. Model checking, software verification, integration into robotic planning and control, and so on.

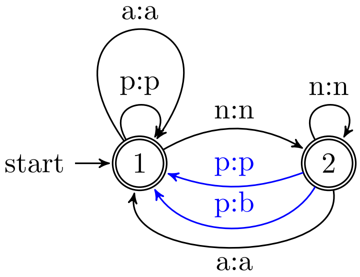
Example #2: ISL Optionality

- For a given k, the k-ISL class of functions is identifiable in the limit in linear time and data.
- Functions are single-valued, no?
- So what about optionality which is rife in natural languages?

(work in progress with Kiran Eiden and Eric Schieferstein)

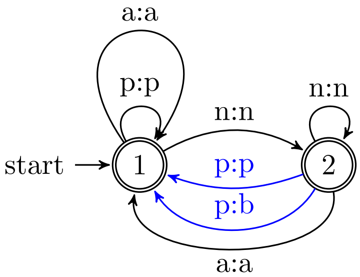
Deterministic FSTs with Language Monoids

Optional Post-nasal Voicing
(Non-deterministic)



Deterministic FSTs with Language Monoids

Optional Post-nasal Voicing (Non-deterministic)

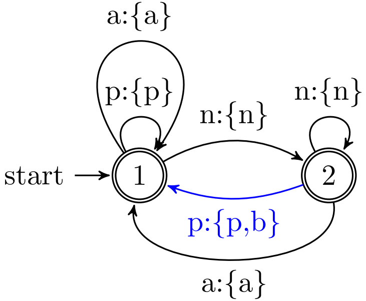


/ a n p a /

$$\begin{tikzcd}
 & & & 1 \arrow[r, "a"] & 1 \\
1 \arrow[r, "a"] & 1 \arrow[r, "n"] & 2 \arrow[ur, "p:b"] \arrow[dr, "p:p"'] & & \\
 & & & 1 \arrow[r, "a"] & 1
\end{tikzcd}$$

Deterministic FSTs with Language Monoids

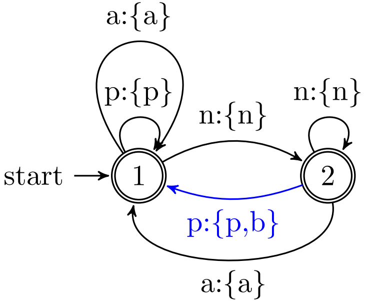
Optional Post-nasal Voicing (Deterministic)



Beros and de la Higuera (2016) call this ‘semi-determinism’.

Deterministic FSTs with Language Monoids

Optional Post-nasal Voicing (Deterministic)

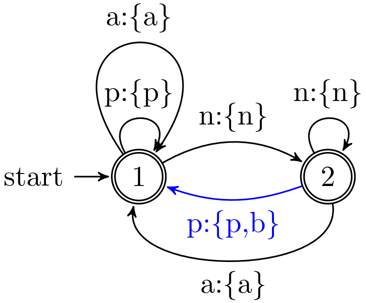


/ a n p a /

$$1 \xrightarrow{\{a\}} 1 \xrightarrow{\{n\}} 2 \xrightarrow{\{p,b\}} 1 \xrightarrow{\{a\}} 1$$

Deterministic FSTs with Language Monoids

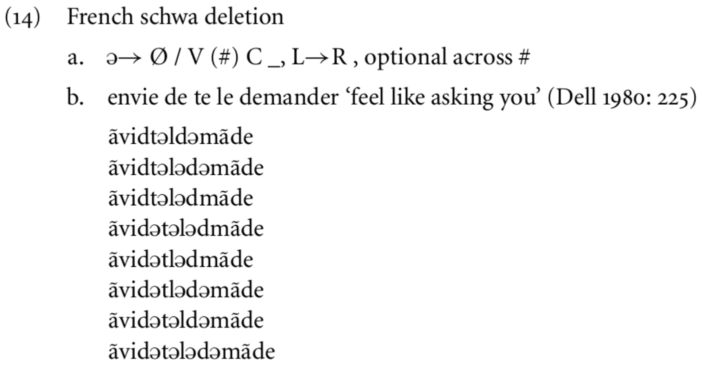
Optional Post-nasal Voicing (Deterministic)



$$/\ a\ n\ p\ a\ / \mapsto \{a\} \cdot \{n\} \cdot \{p, b\} \cdot \{a\} = \{anpa, anba\}$$

Iterative Optionality is more challenging

Vaux 2008, p. 43



(14) French schwa deletion

a. ə→ Ø / V (#) C _, L→R , optional across #

b. envie de te le demander ‘feel like asking you’ (Dell 1980: 225)

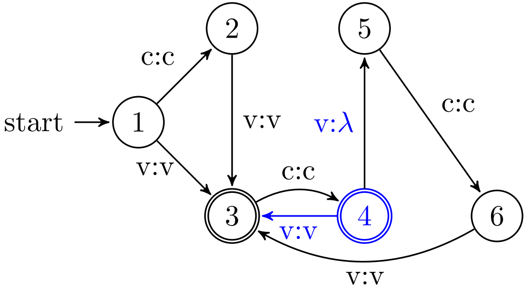
ãvidtəldəmãde
ãvidtələdəmãde
ãvidtələdmãde
ãvidətələdmãde
ãvidətlədmãde
ãvidətlədəmãde
ãvidətəldəmãde
ãvidətələdəmãde

Abstract Example

V $\rightarrow \varnothing$ / VC __ CV (applying left-to-right)

/cvcv/	/cvcvcv/	/cvcvcvcv/	/cvcvcvcvcv/	
cvcv	cvcvcv	cvcvcvcv	cvcvcvcvcv	faithful
	cvccv	cvccvcv	cvccvcvcv	2nd vowel deletes
		cvcvccv	cvcvccvcv	3rd vowel deletes
			cvccvccv	2nd, 4th vowels delete

Problem: Output-oriented Optionality



- The output determines the state!
- $4 \xrightarrow{\text{v:}\{\text{v},\lambda\}} ??$
- For **deterministic** transducers, the next state is necessarily determined by the input symbol!

What would Kisseberth say?

Kisseberth 1970: 304-305

By making …rules meet two conditions (one relating to the form of the input string and the other relating to the form of the output string; one relating to a single rule, the other relating to all the rules in the grammar), we are able to write the vowel deletion rules in the intuitively correct fashion. We do not have to mention in the rules themselves that they cannot yield unpermitted clusters. We state this fact once in the form of a derivational constraint.

Learn the ISL function and surface constraints independently and simultaneously

Strategy

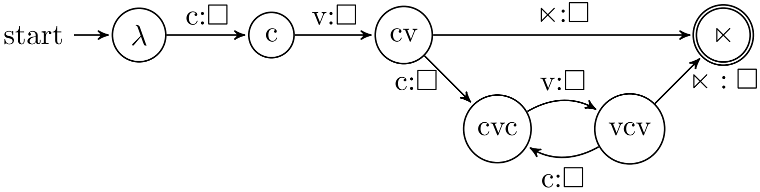
Learn an Input-based function (T_1) and filter the outputs with phonotactic constraints (T_2).

$$T_1 \circ T_2 = \text{target}$$

Learning the ISL function

- Algorithm synthesizes aspects of Jardine et al. (2014) and Beros and de la Higuera (2016).

Before Learning

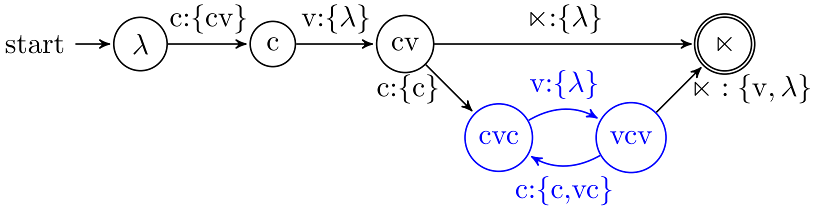


Input Strictly Local Transducer with 4-size window

Learning the ISL function

- Learns to optionally delete every vowel except the 1st!!

After Learning



Summary of this example

1. Optional processes can be deterministic.
   (Multi-valued $\not\rightarrow$ non-deterministic.)
   Deterministic String Relations

Summary of this example

1. Optional processes can be deterministic.
   (Multi-valued $\not\rightarrow$ non-deterministic.)
   Deterministic String Relations
2. Non-decomposed, output-oriented, optional processes are non-deterministic.

Summary of this example

1. Optional processes can be deterministic.
   (Multi-valued $\not\rightarrow$ non-deterministic.)
   Deterministic String Relations
2. Non-decomposed, output-oriented, optional processes are non-deterministic.
3. But they can be factored into a deterministic process which overgenerates and a constraint which filters out the unwanted overgenerates.

$$T = T_1 \circ T_2$$

Summary of this example

1. Optional processes can be deterministic.
   (Multi-valued $\not\rightarrow$ non-deterministic.)
   Deterministic String Relations
2. Non-decomposed, output-oriented, optional processes are non-deterministic.
3. But they can be factored into a deterministic process which overgenerates and a constraint which filters out the unwanted overgenerates.

$$T = T_1 \circ T_2$$

4. T_2 can be learned with existing grammatical inference methods.

Summary of this example

1. Optional processes can be deterministic.
(Multi-valued $\not\rightarrow$ non-deterministic.)
Deterministic String Relations
2. Non-decomposed, output-oriented, optional processes are non-deterministic.
3. But they can be factored into a deterministic process which overgenerates and a constraint which filters out the unwanted overgenerates.

$$T = T_1 \circ T_2$$

4. T_2 can be learned with existing grammatical inference methods.
5. T_1 appears to be learnable with a synthesis of recent results in grammatical inference.

Discussion

1. Formal proof of correctness of the algorithm for learning classes of structured multi-valued functions is in progress.

Discussion

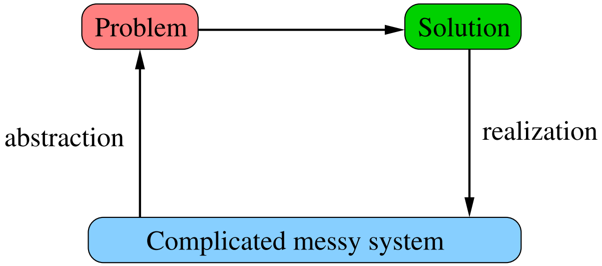
1. Formal proof of correctness of the algorithm for learning classes of structured multi-valued functions is in progress.
2. Probabilities can be appended to the outputs for learning classes of functions.

Discussion

1. Formal proof of correctness of the algorithm for learning classes of structured multi-valued functions is in progress.
2. Probabilities can be appended to the outputs for learning classes of functions.
3. We hope to apply to other problems:
    1. learning URs and phonological grammars simultaneously
    2. sociolinguistic variation
    3. NLP problems such as G2P, P2G, and so on.

Discussion

1. Formal proof of correctness of the algorithm for learning classes of structured multi-valued functions is in progress.
2. Probabilities can be appended to the outputs for learning classes of functions.
3. We hope to apply to other problems:
   1. learning URs and phonological grammars simultaneously
   2. sociolinguistic variation
   3. NLP problems such as G2P, P2G, and so on.



Part V

A Summary of Sorts

Personal View

Two seeds in the 20th century

1. Mathematics can be developed to provide stronger/tighter characterizations of natural language patterns.
2. Computational Learning Theory stresses the importance and necessity of structured hypothesis spaces. Don't treat them cavalierly!

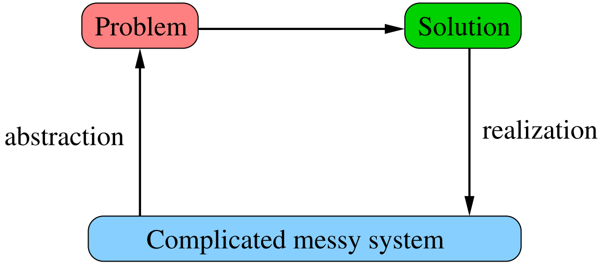
These compatible ideas are bearing fruit well into the 21st century.

More 21st century Highlights

1. Abstract Categorial Grammars (De Groote 2001)
2. The Syntactic Concept Lattice (Clark 2013)
3. The Tolerance Principle (Yang 2016)
4. . . .

Conclusion

1. Far from being a fossil from a former era, mathematical thinking about language
    1. continues to play an essential role in understanding natural languages and
    2. continues to make critical contributions to our understanding of how things which compute—both humans and machines—can learn.



The End

Thanks for listening, and thanks to everyone I have ever spoken with: students, mentors and peers.

Let's discuss more on the Outdex!

`https://outde.xyz/`