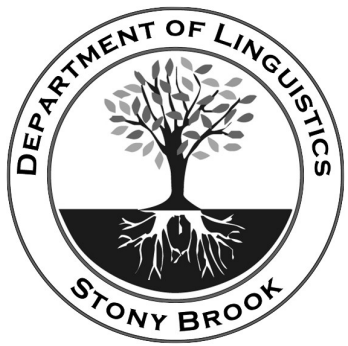
Seminar: Foundations of Mathematical and Computational Linguistics

Jeffrey Heinz



Stony Brook University

V-NYI

2026/01/09

Welcome

I  N Y

Welcome

I 

V N Y I

Instructors

- Jeffrey Heinz is a professor of Linguistics at Stony Brook University in New York.
- Vinny Czarnecki is a PhD student at Rutgers University in New Jersey.

This Course

Mathematical and computational thinking about language plays an essential role in understanding natural languages, which provides insights into:

- the limits and kinds of cross-linguistic variation
- language processing
- learning and acquisition

Outline

- What is mathematical linguistics? Computational linguistics?
- Overview of this course
- Examples of grammatical formalisms

Part I

What is mathematical linguistics?

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

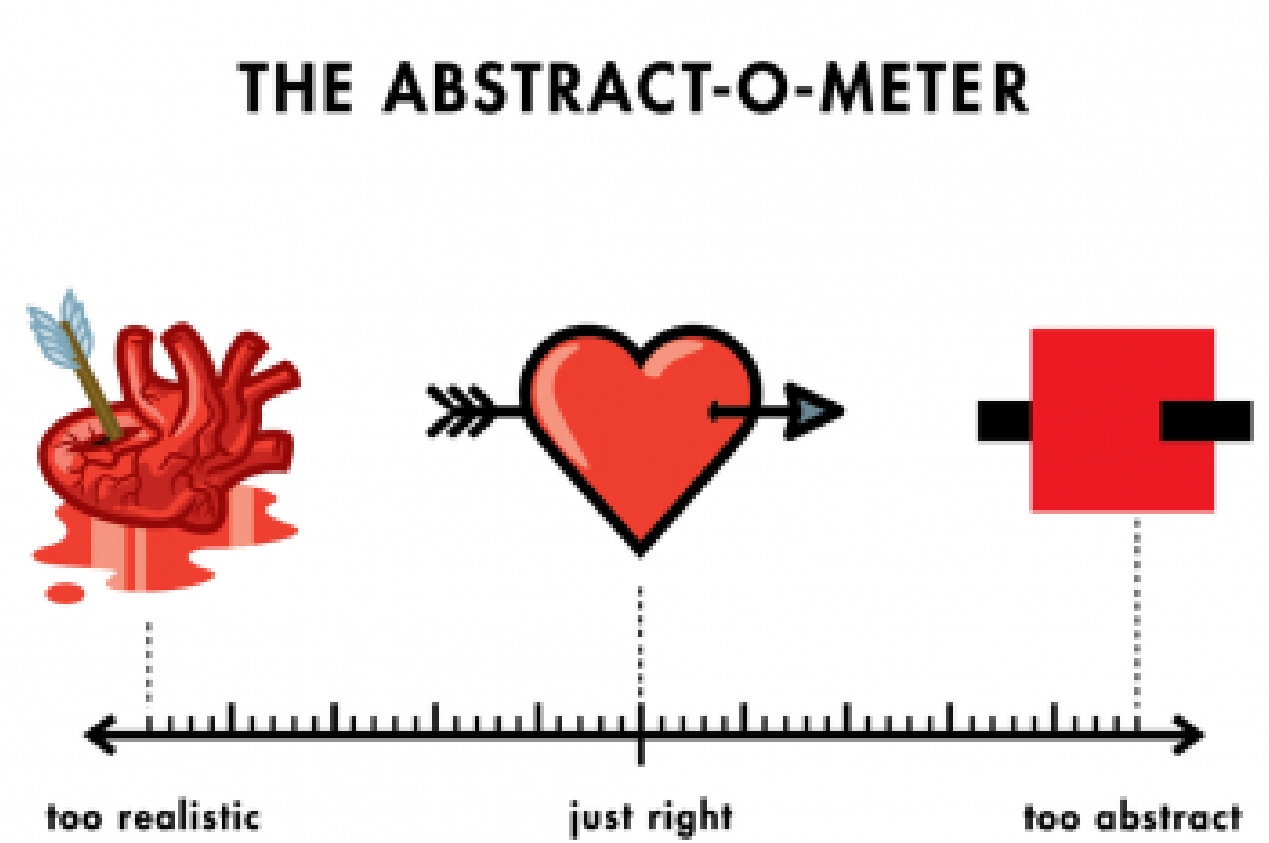
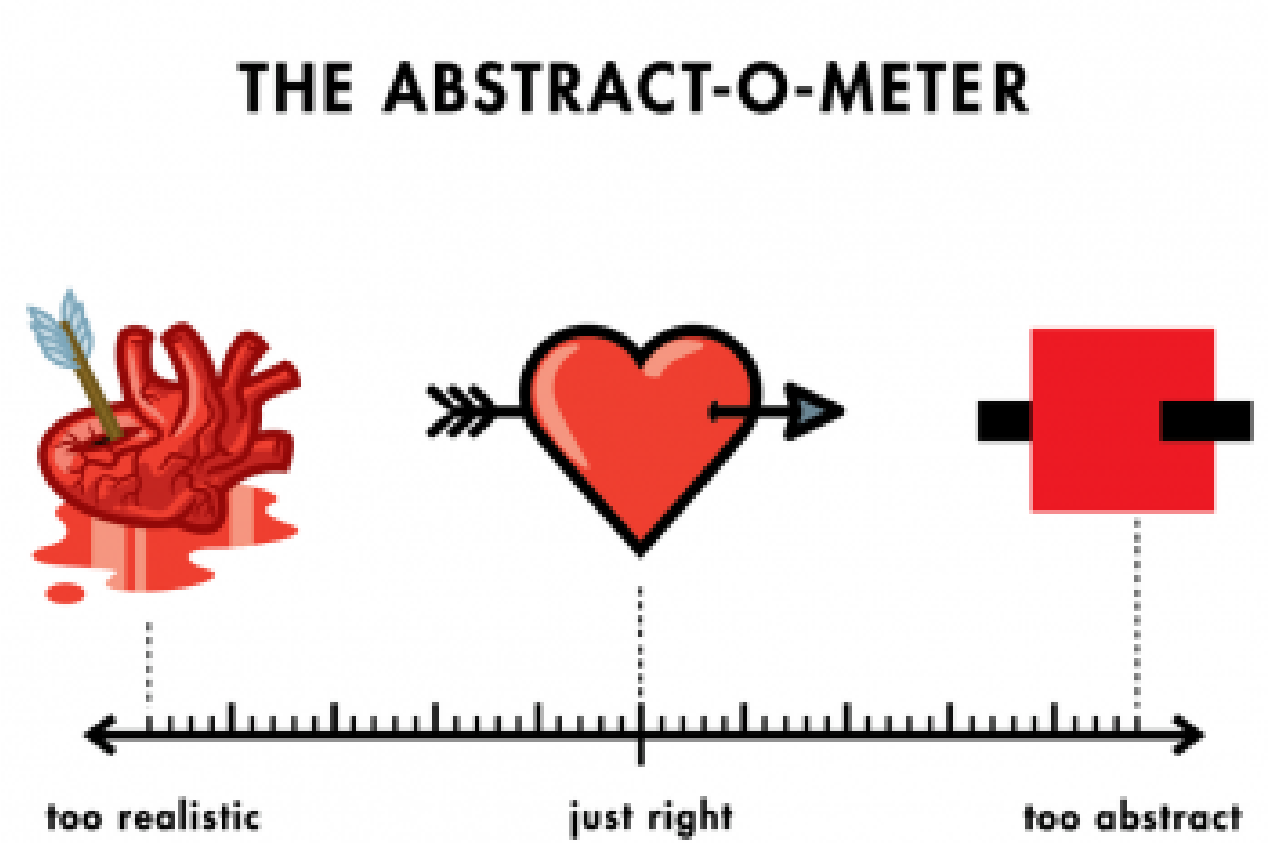


image credit: https://computersciencewiki.org/index.php/Abstraction

Abstraction



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

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

Part II

What is computational linguistics?

What is computer science?

Edsger W. Dijkstra (folklore)

"Computer science is no more about computers than astronomy is about telescopes."

Algorithms

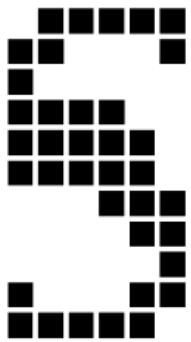
An algorithm is a set of steps to accomplish a task.

Thomas H. Cormen (2013)

"Computer algorithms solve computational problems. We want two things from a computer algorithm: given an input to a problem, it should always produce a correct solution to the problem, and it should use computational resources efficiently while doing so."

Correctness (Cormen 2013)

1. We often specify precisely what a correct solution would entail.
2. Sometimes we cannot. Is this 11×6 pixel image a 5 or an S?



3. Tricky Cases:
   1. Sometimes it is accetable for an algorithm to be correct "most of the time."
   2. Sometimes it is acceptable for an algorithm to be "approximately correct."

Resource Usage (Garey and Johnson 1979)

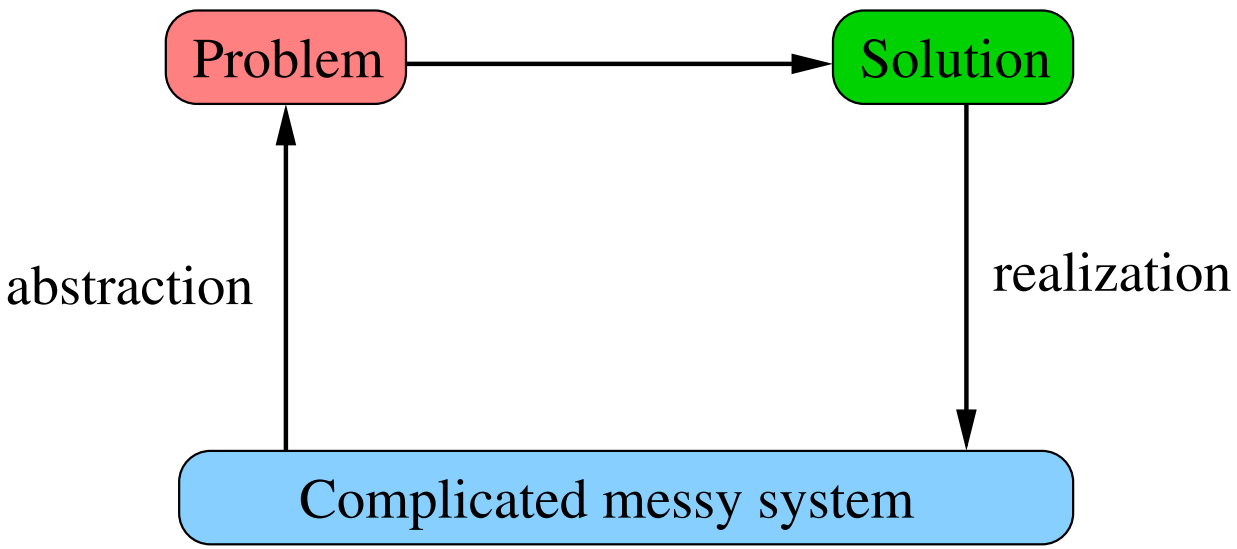
Size of Largest Problem Instance
Solvable in 1 Hour

Time complexity function	With present computer	With computer 100 times faster	With computer 1000 times faster
n	N_1	$100\ N_1$	$1000\ N_1$
n^2	N_2	$10\ N_2$	$31.6\ N_2$
n^3	N_3	$4.64\ N_3$	$10\ N_3$
n^5	N_4	$2.5\ N_4$	$3.98\ N_4$
2^n	N_5	$N_5+6.64$	$N_5+9.97$
3^n	N_6	$N_6+4.19$	$N_6+6.29$

Figure 1.3 Effect of improved technology on several polynomial and exponential time algorithms.

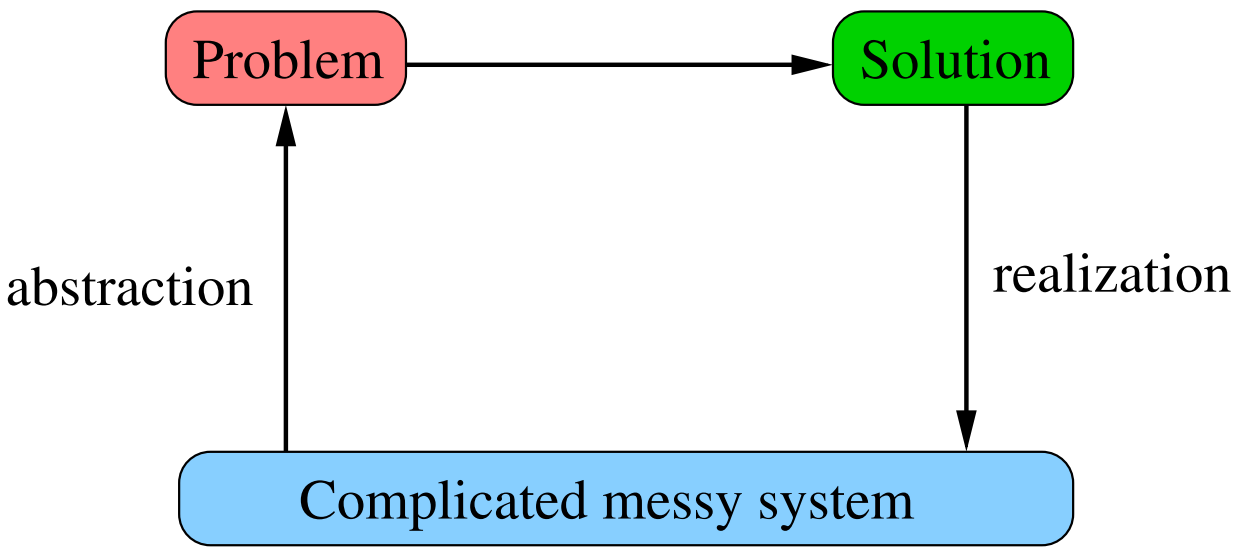
Advantages:

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



Disadvantages:

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



Part III

Mathematics, Computer Science and Language

Mathematics of Sequences

- Language unfolds over time.
- We observe sequences of linguistic events.
- What is the mathematics of sequences?

Mathematics of Sequences

- Language unfolds over time.
- We observe sequences of linguistic events.
- What is the mathematics of sequences?

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

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

Mathematics of Sequences

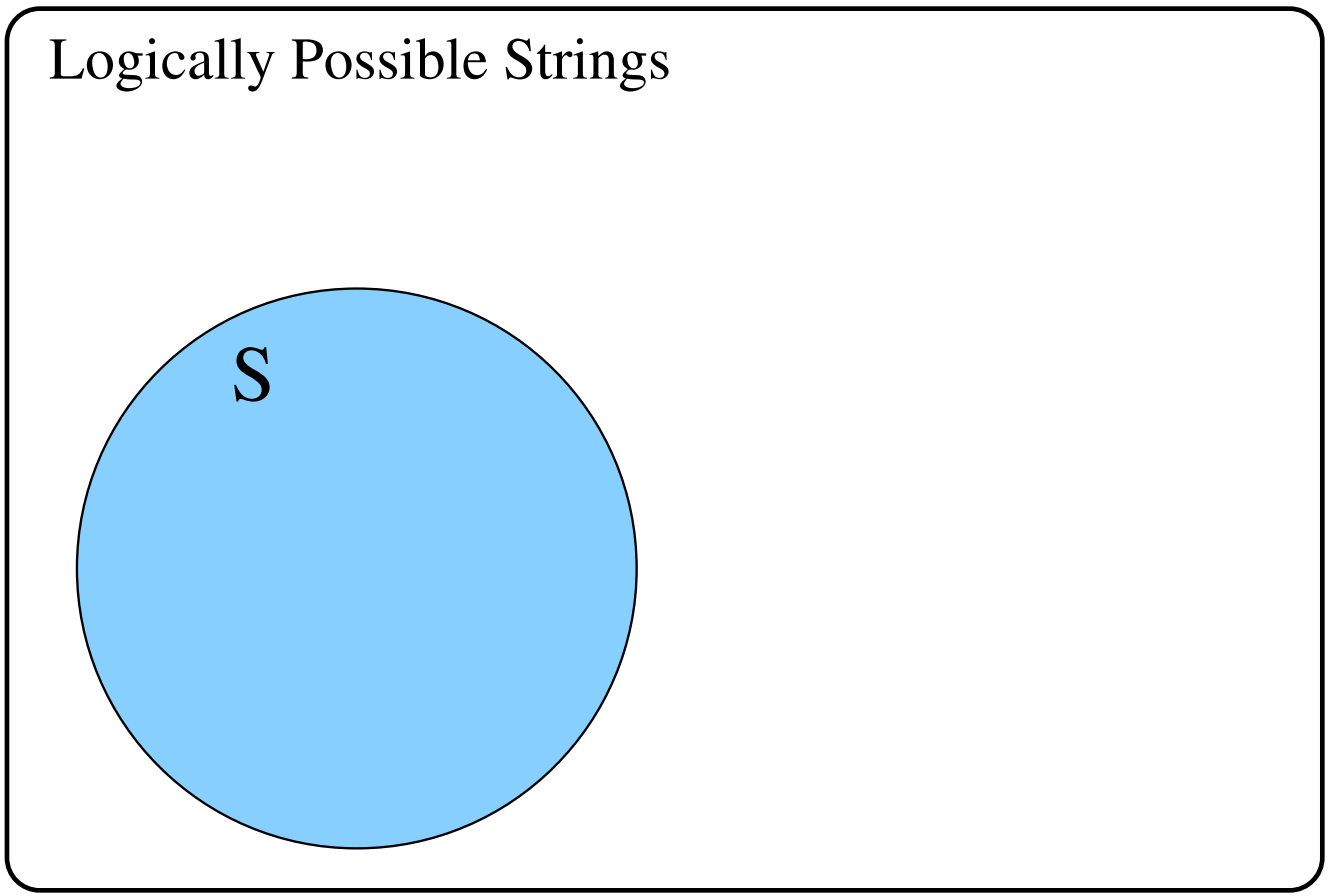
- Language unfolds over time.
- We observe sequences of linguistic events.
- What is the mathematics of sequences?

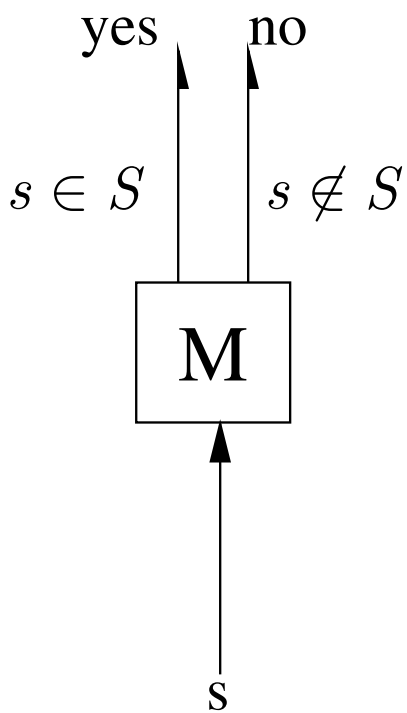
Knowledge of language includes knowledge of which sequences are licit and which are not.

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

Of course it includes much more as well!

A Membership problem





Detour to Czarnecki and Nelson section 1.3-1.5

Let's look at

1. intensional and extensional descriptions of sets.
2. relations (briefly)
3. functions (briefly)

Variations thereof

Functions on the domain of strings ...

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 domain of strings …

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 domain of strings …

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, …

How can we classify functions like those above?

Variations thereof

Functions on the domain of strings …

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, …

How can we classify functions like those above?

What about functions whose domain is tree structures?

Doing Linguistic Typology

Requires two books:

- "encyclopedia of categories"
- "encyclopedia of types"

Wilhelm Von Humboldt

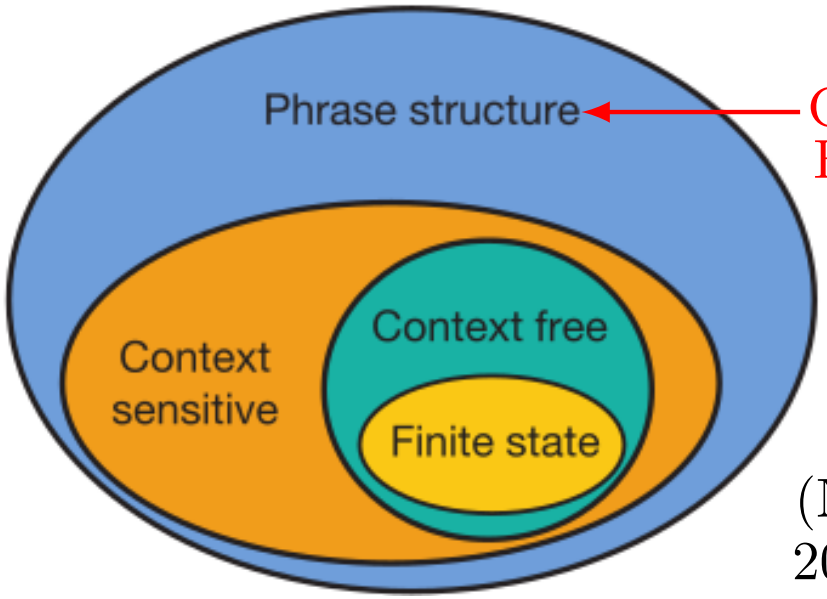
Doing Linguistic Typology

Requires two books:

- "encyclopedia of categories"
- "encyclopedia of types"

Formal grammars provide a mathematical, computational encyclopedia of categories.

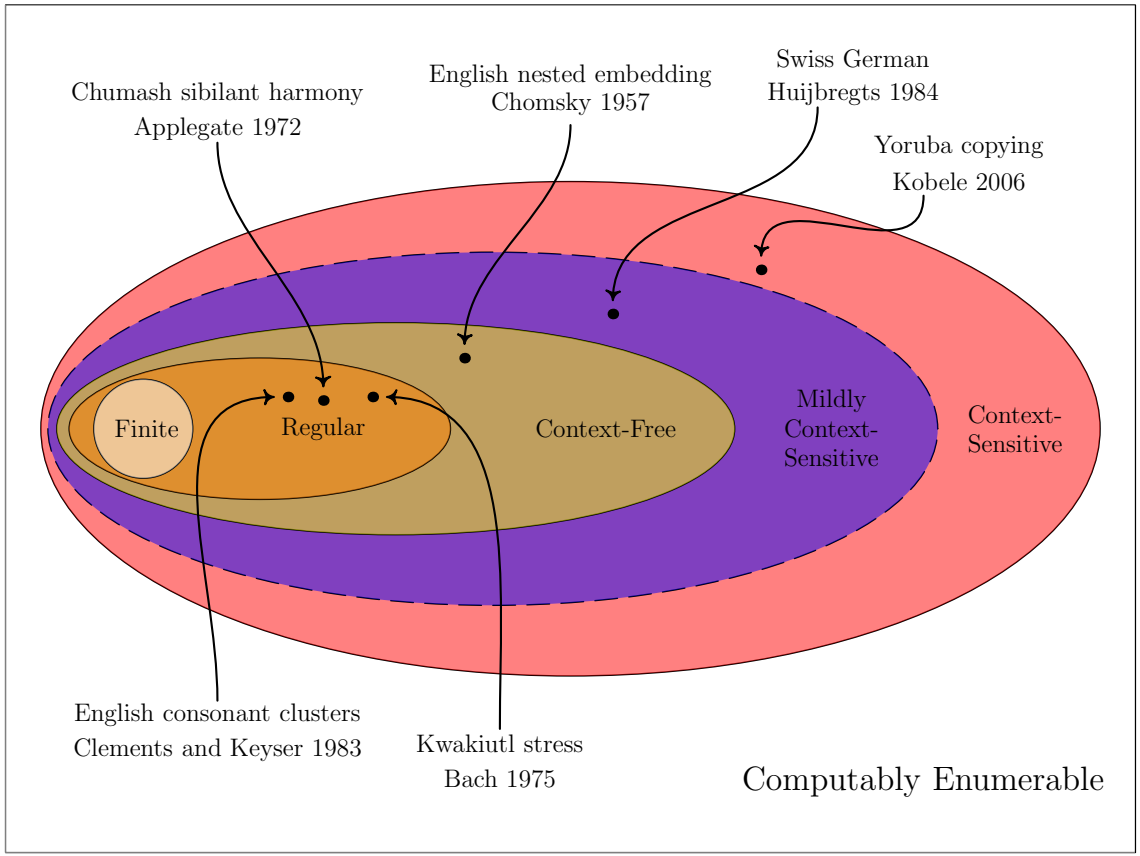
Classifying Membership Problems (20th century view)



(Nowak et al. 2002, *Nature*)

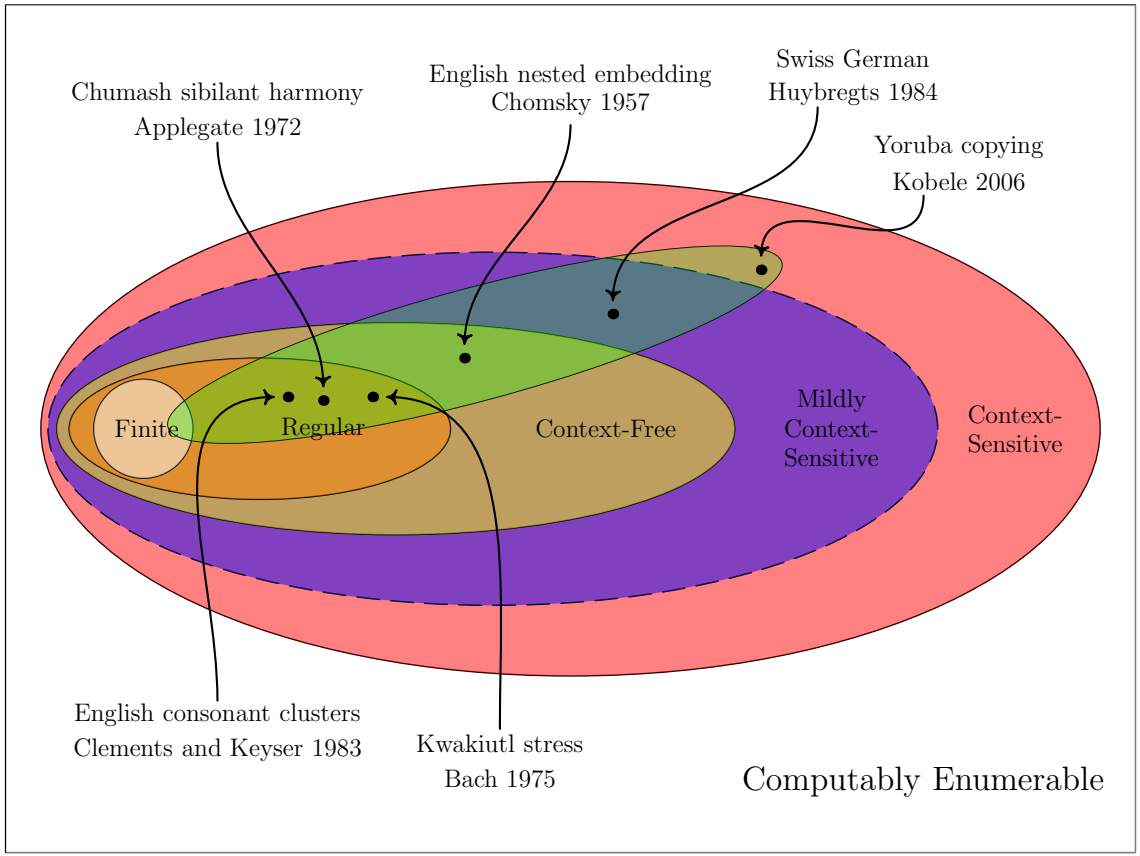
Figure 3 The Chomsky hierarchy and the logical necessity of universal grammar. Finite-state grammars are a subset of context-free grammars, which are a subset of context-sensitive grammars, which are a subset of phrase-structure grammars, which represent all possible grammars. Natural languages are considered 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.

Where is natural language? (20th century view)



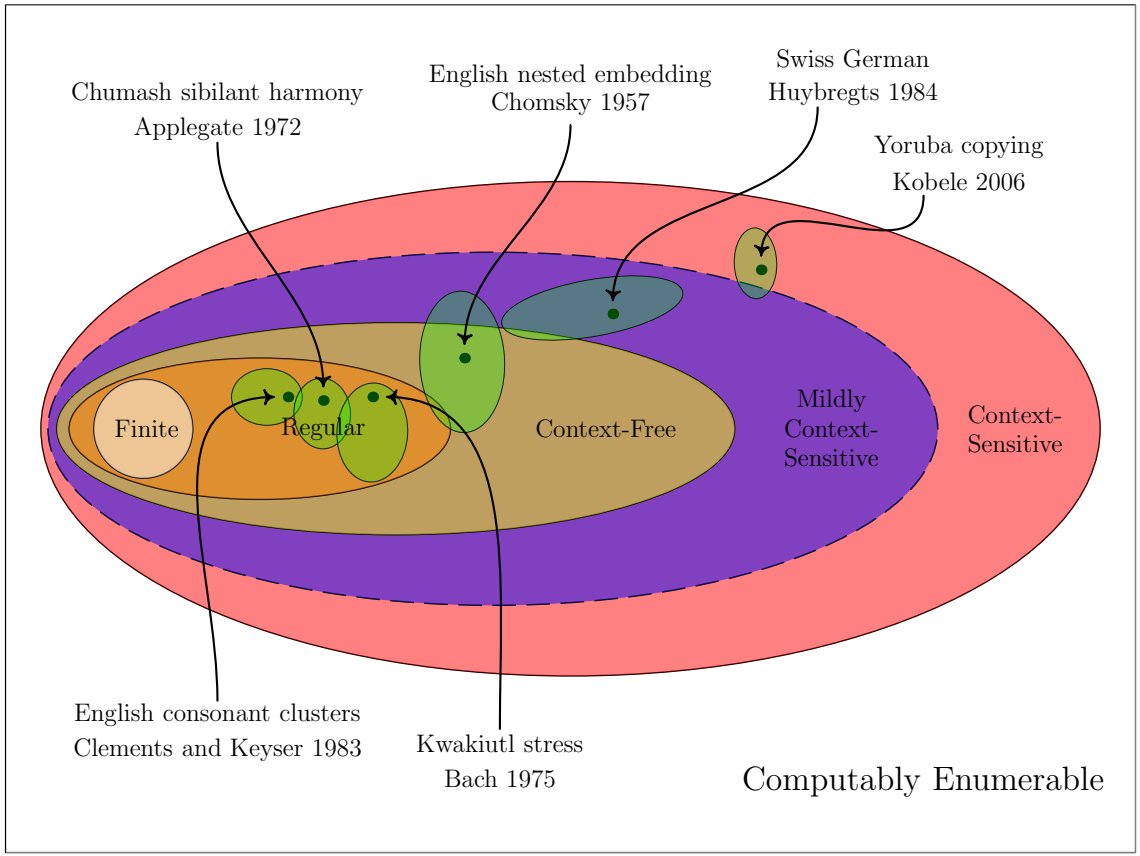
1. Morphology/phonology is regular with the exception of total reduplication
2. Syntax not regular and not even context-free

Where is natural language? (20th century view)



1. Morphology/phonology is regular with the exception of total reduplication
2. Syntax not regular and not even context-free

Where is natural language? (20th century view)



1. Morphology/phonology is regular with the exception of total reduplication
2. Syntax not regular and not even context-free

"Regular" Computations

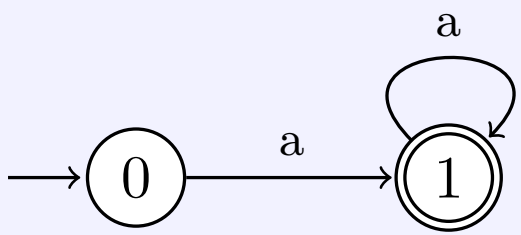
Logical formula

$$(\forall x)[\mathtt{a}(x)] \wedge (\exists x)[\mathtt{a}(x)]$$

Regular expression

$$aa^*$$

Finite-state acceptor



For each type of grammar, there is an effective procedure which checks whether any string satisfies the expression.

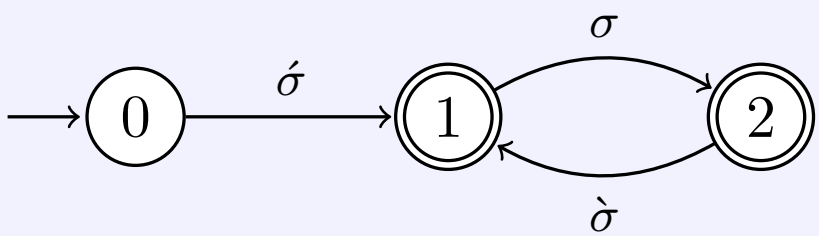
Maranguku Stress

σ́σ	σ́σσ̀σσ̀	
σ́σσ̀	σ́σσ̀σσ̀σ	
σ́σσ̀σ	σ́σσ̀σσ̀σσ̀	...

Regular expression

$$\acute{\sigma}(\sigma\grave{\sigma})^* + \acute{\sigma}(\sigma\grave{\sigma})^*\sigma$$

Finite-state acceptor



Chandlee and Heinz (2017)

Maranguku Stress

σ́σ	σ́σσ̀σσ̀	
σ́σσ̀	σ́σσ̀σσ̀σ	
σ́σσ̀σ	σ́σσ̀σσ̀σσ̀	...

$$
\begin{aligned}
\texttt{first}(x) &\overset{\text{def}}{=} \neg(\exists y)[y \lhd x] \\
\texttt{stress}(x) &\overset{\text{def}}{=} \acute{\sigma}(x) \vee \grave{\sigma}(x) \\
\rtimes\acute{\sigma} &\overset{\text{def}}{=} (\exists x)[\acute{\sigma}(x) \wedge \texttt{first}(x)] \\
\texttt{x}\acute{\sigma} &\overset{\text{def}}{=} (\exists x)[\acute{\sigma}(x) \wedge \neg\texttt{first}(x)] \\
\texttt{lapse} &\overset{\text{def}}{=} (\exists x, y)[x \lhd y \wedge \sigma(x) \wedge \sigma(y)] \\
\texttt{clash} &\overset{\text{def}}{=} (\exists x, y)[x \lhd y \wedge \texttt{stress}(x) \wedge \texttt{stress}(y)]
\end{aligned}
$$

Logical formula

$$\texttt{Maranungku} \overset{\text{def}}{=} \rtimes\acute{\sigma} \wedge \neg\texttt{x}\acute{\sigma} \wedge \neg\texttt{lapse} \wedge \neg\texttt{clash}$$

Chandlee and Heinz (2017)

Nonregularity of Syntax

$$a^n b^n = \{ab, aabb, aaabbb, \ldots\}.$$

Example 1 (English)

1. oysters eat
2. oysters oysters eat eat
3. ...

Example 2 (Swiss German)

... that we the children Hans the house let help paint

... das mer d'chind em Hans es huus lönd hälfe aastriiche

- N N N V V V

Part III

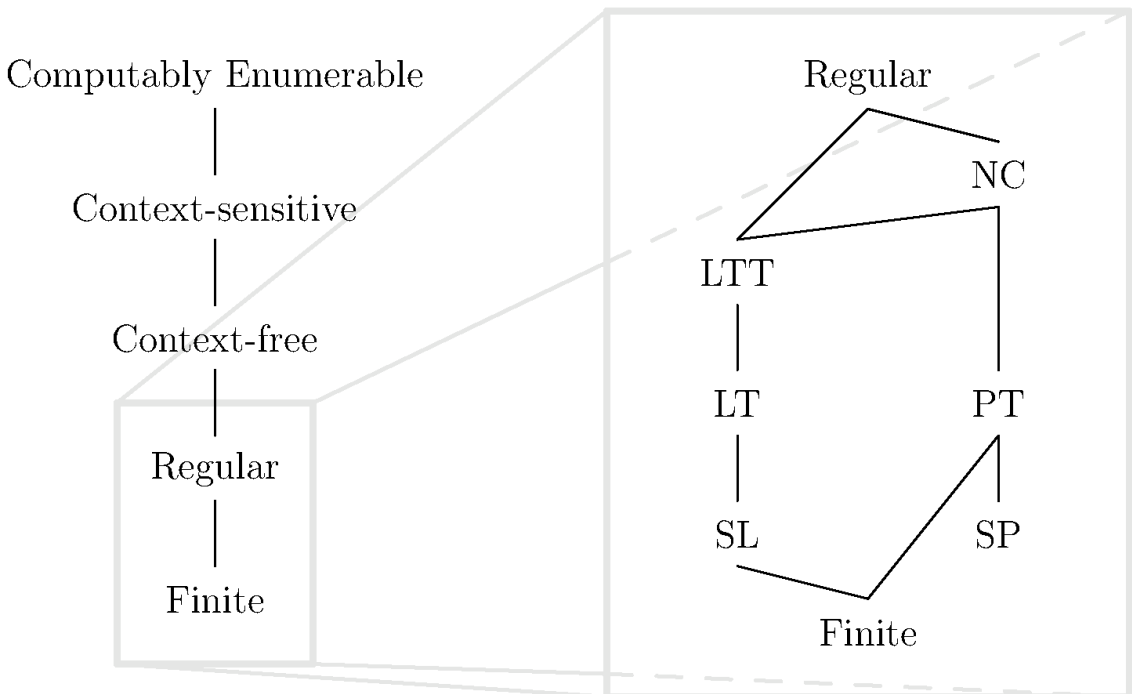
The 21st Century (Present)

Advances

1. **Granularity** of Characterizations
    1. Sets of sequences: $\Sigma^* \rightarrow \{0, 1\}$
    2. Sequence to sequence transformations: $\Sigma^* \rightarrow \Delta^*$
2. **Unification** of Computational Analysis of Syntax/Phonology/Morphology via Representations (Trees and Sequences).
3. **Connections** of These Characterizations to Computational Learning Theory.

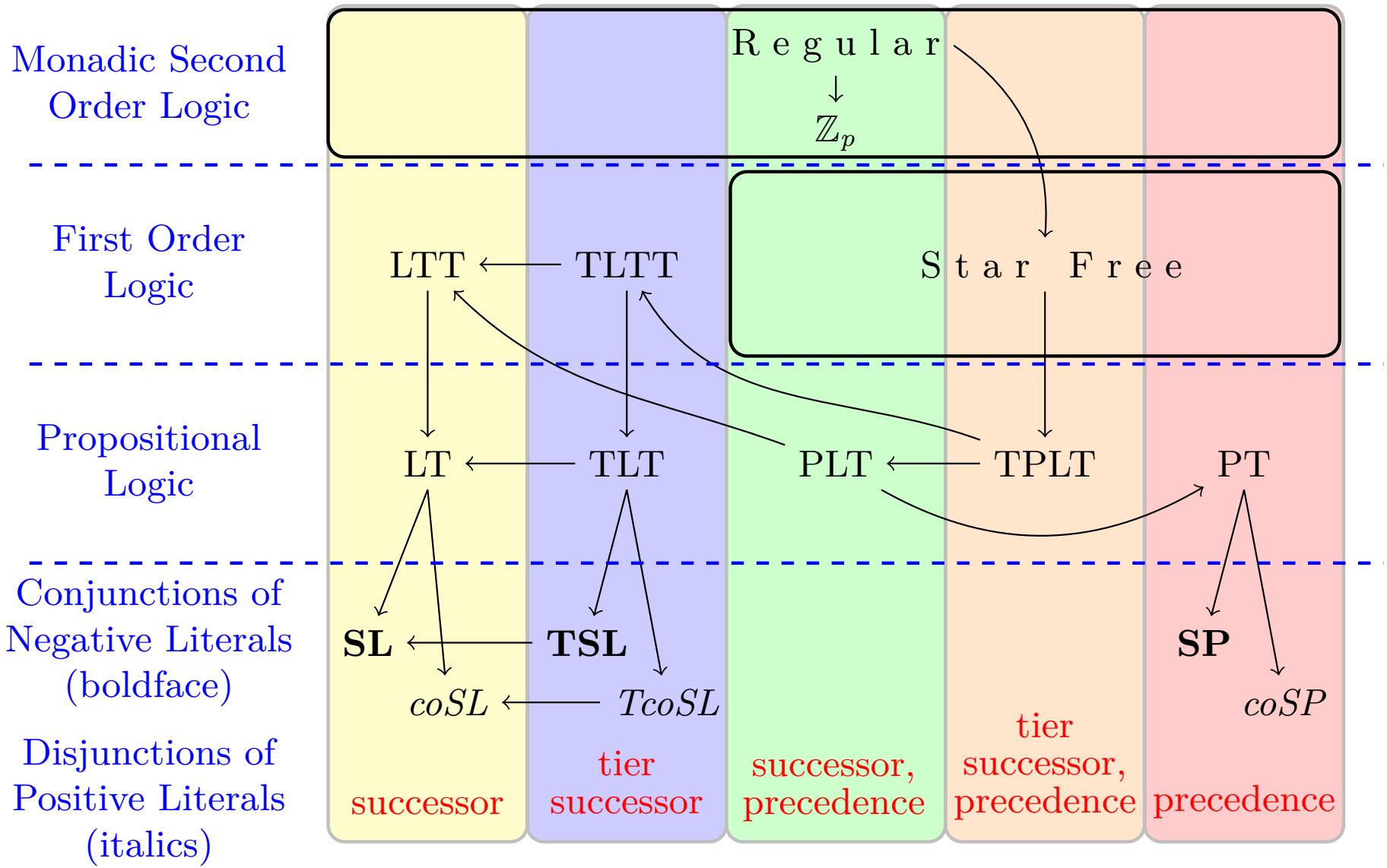
Granularity of Characterizations

- The Regular/Context-Free/Context-Sensitive distinctions are **course-grained**.
- We now have much **finer-grained** characterizations.



- Strikingly, these subregular classes, when applied to trees, let us probe the non-regular stringsets!

Example: Subregular Sets of Sequences



Phonotactic patterns

1. Nearly all are at the bottom!

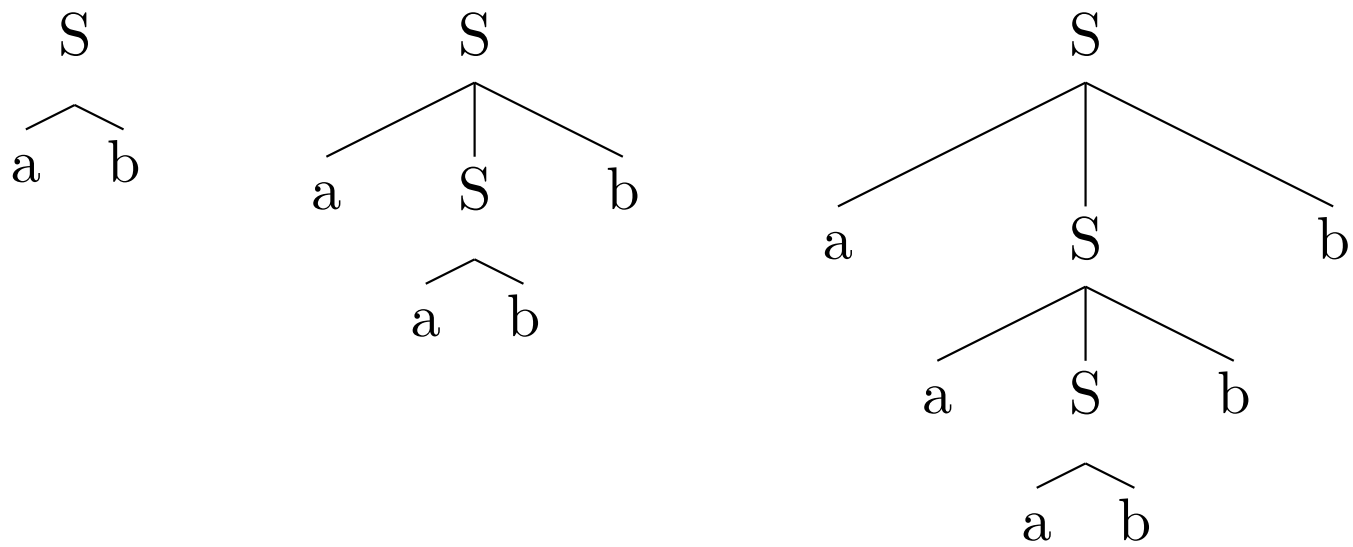
   SL, coSL, TSL, TcoSL, SP, coSP

2. And are consquently describable with simple logical languages and well-behaved automata.
3. These classes are "local" in particular, precise ways.
4. These classes are sufficiently structured to facilitate their learnability (with a parameter k).

(Heinz 2018, Lambert et al. 2021, van der Poel et al. 2024, Lambert, to appear)

Logic and Finite-State Automata handle trees too!

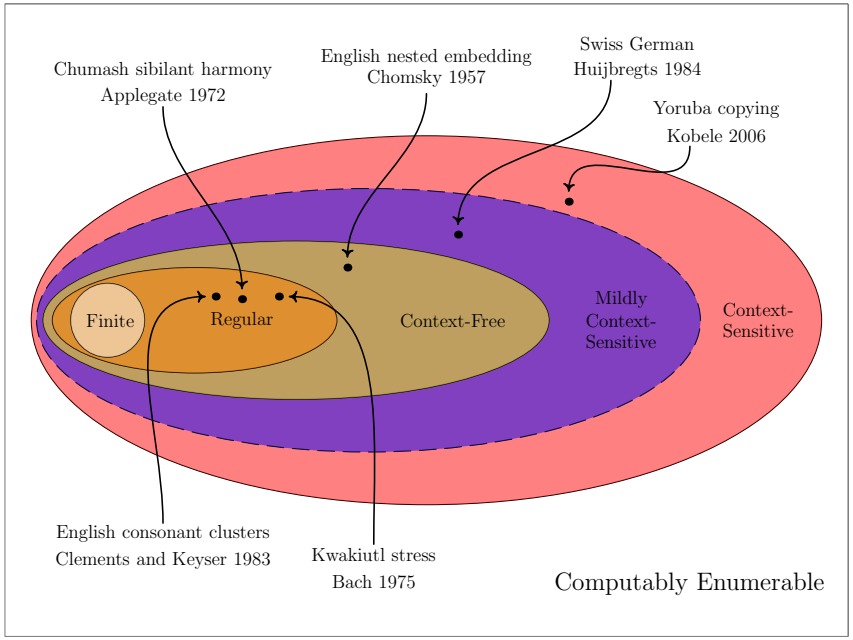
Tree structures are a foundation to distinct theories of syntax (Stabler 2019).



...

These trees spell-out the sequences ab, $aabb$, $aaabbb$ respectively.

Mildly Context Sensitive Stringsets



1. Take a set of trees T recognized by a finite-state tree acceptor.

$$T$$

2. Transform the trees in T with a finite-state tree transducer F.

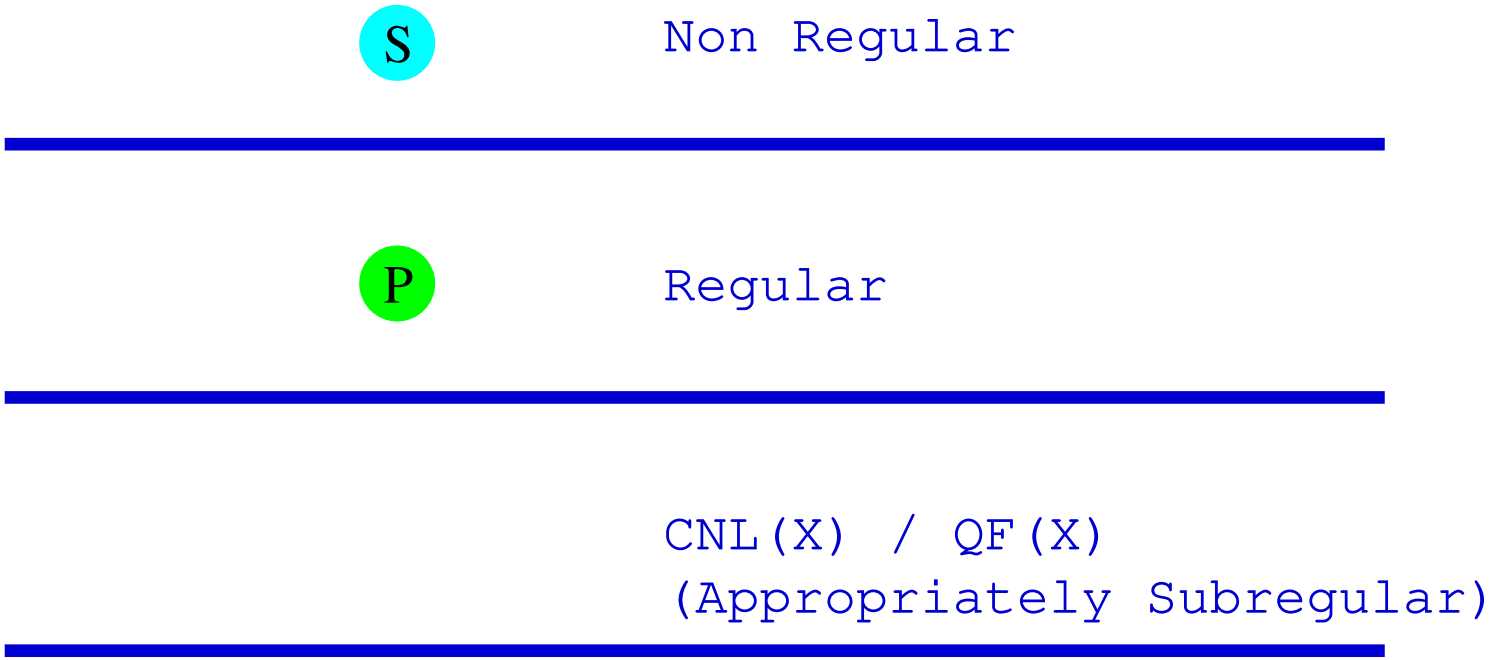
$$F(T)$$

3. Spell-out those trees to yield a set of strings.

$$Y(F(T))$$

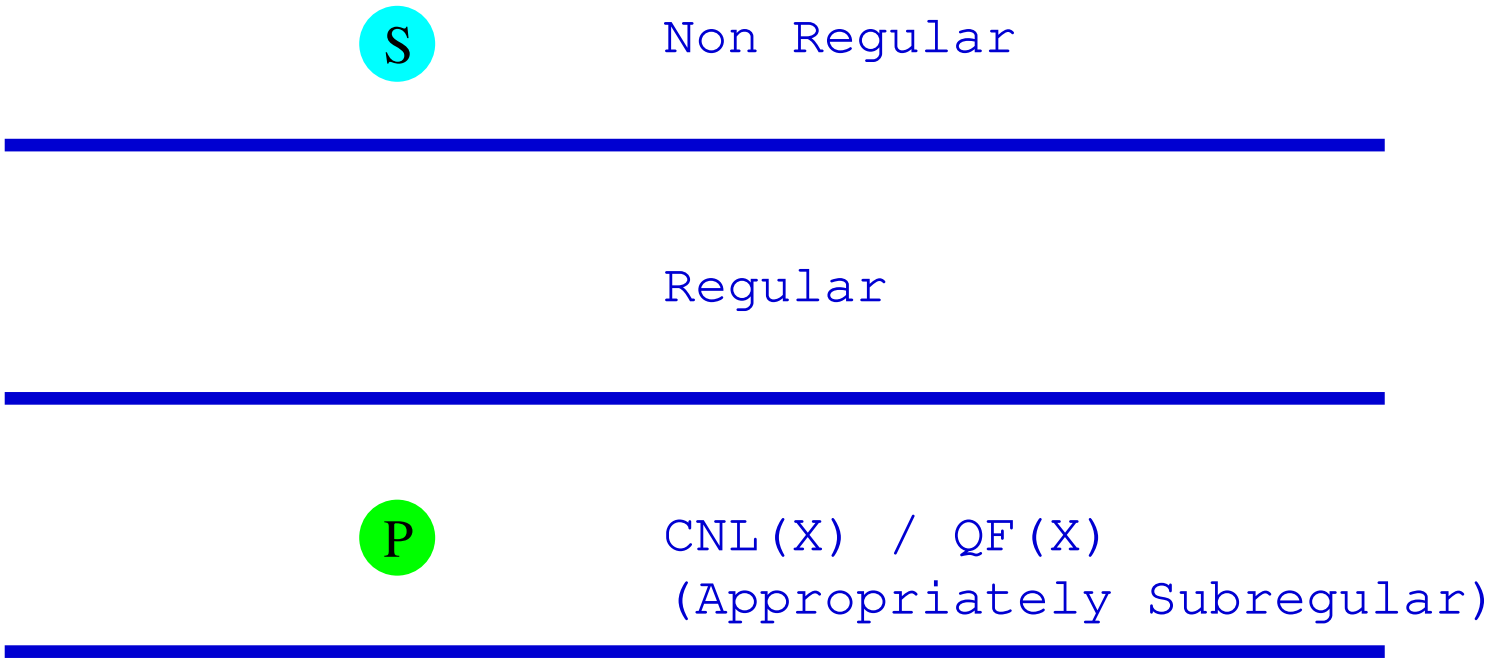
4. That set of strings is mildly context-sensitive (like Swiss German)!

Subregular Syntax



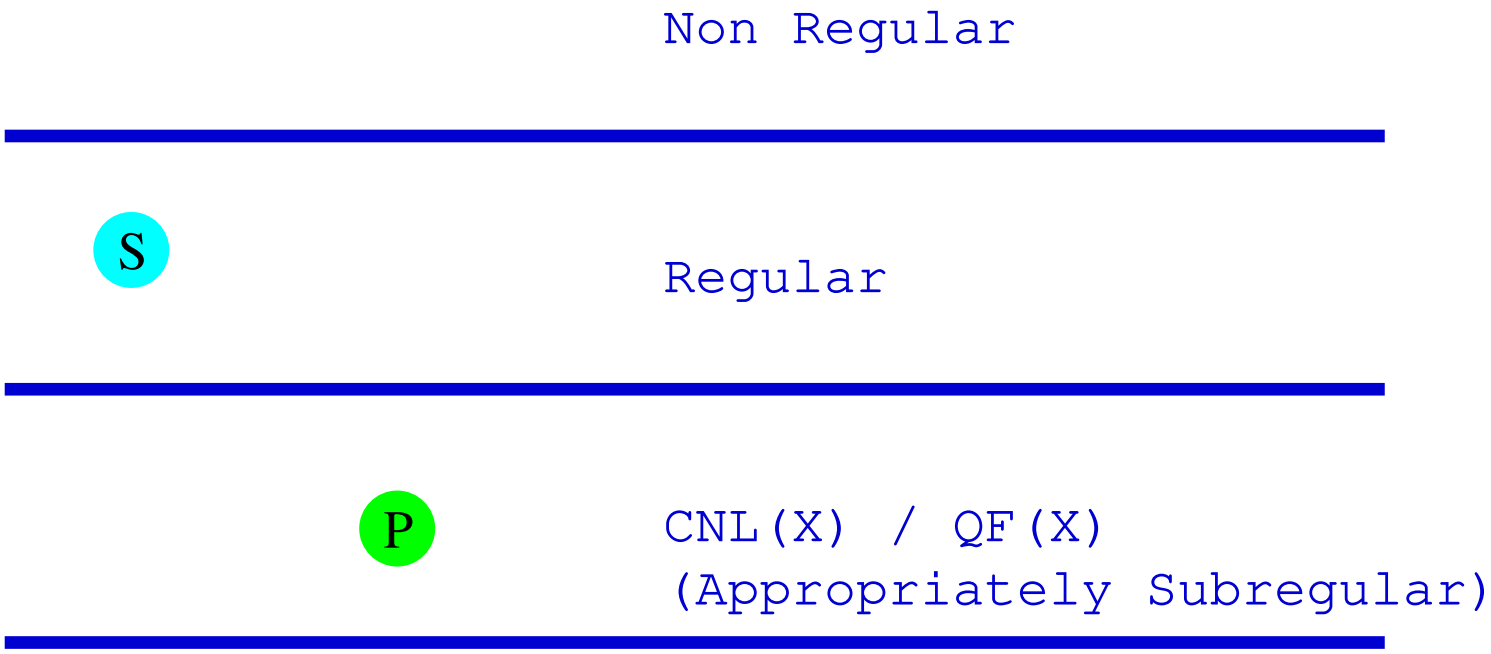
20th century view

Subregular Syntax



(Heinz 2018)

Subregular Syntax



(Stabler 2019)

Subregular Syntax

Non Regular

Regular

S P CNL(X) / QF(X)
(Appropriately Subregular)

---

trees strings

(Graf 2022)

Rest of the day today

1. Formal Grammars I: Regular Expressions and Variations thereof
2. Formal Grammars II: Propositional and First Order Logic