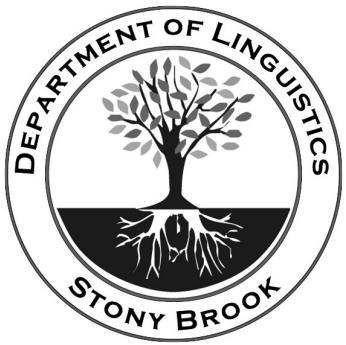
What does learning mean?

Jeffrey Heinz



AI Institute
Stony Brook University
February 6, 2020

What does Learning Mean?

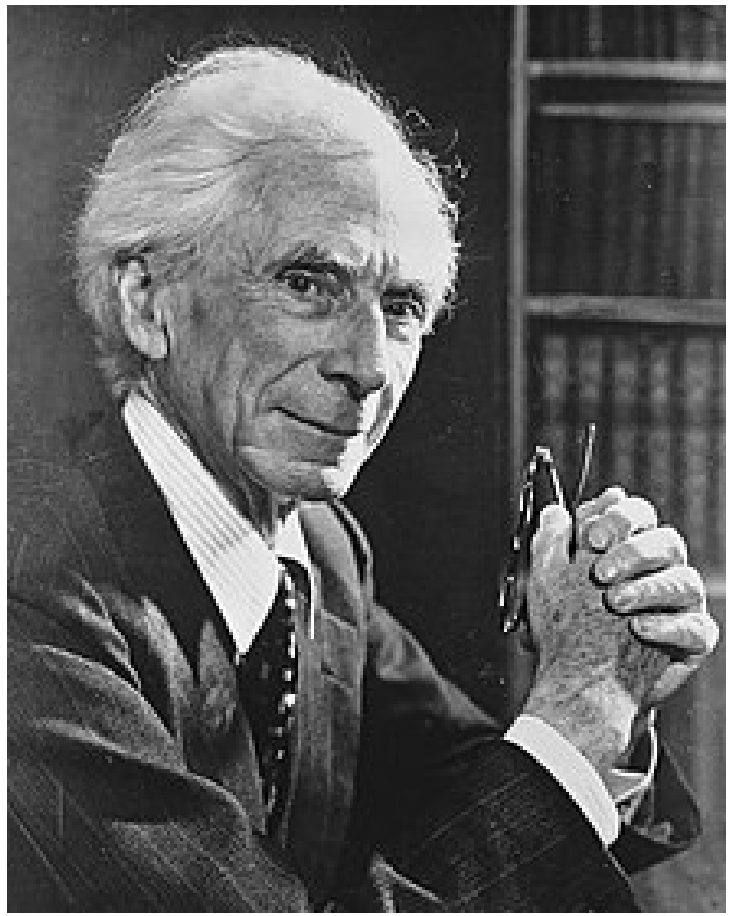
- Personal Motivation
- A Tour of Computational Learning Theory
- . . . and consequences thereof
- Examples along this other path

Part I

Personal Motivation

What does Learning Mean?

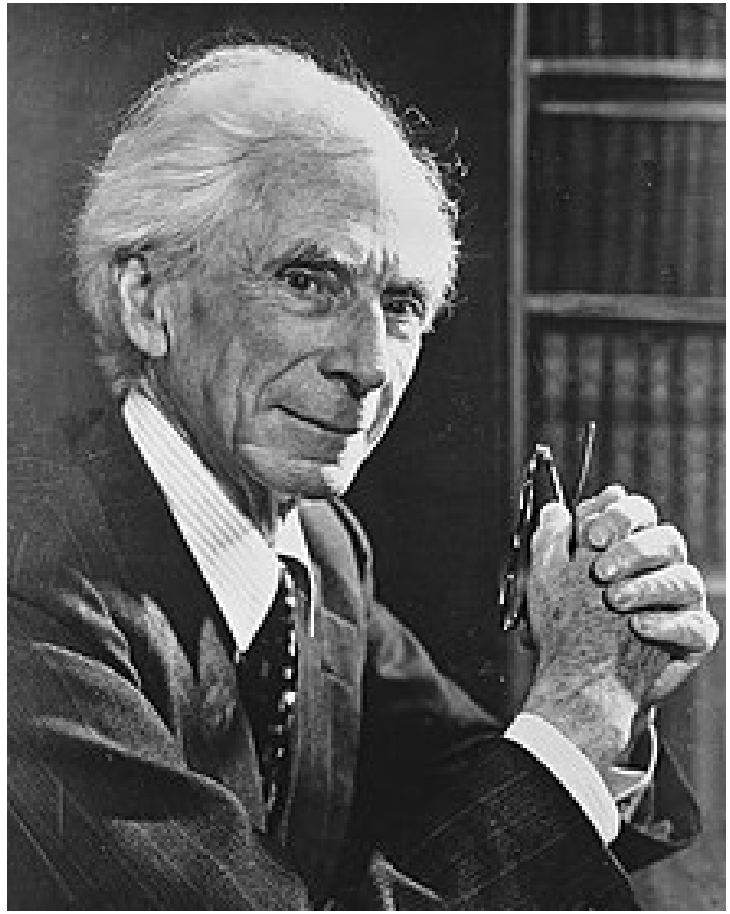
How comes it that human beings, whose contacts with the world are brief and personal and limited, are nevertheless able to know as much as they do know? (1935)



Bertrand Russell (circa 1957)

What does Learning Mean?

. . . if we are to be able to draw inferences from these data. . . we must know . . . principles of some kind by means of which such inferences can be drawn. (1912)



Bertrand Russell (circa 1957)

One thing we know is our own language

[Play video of Ella]

Learning Language is Deceptively Hard

The Human Speechome Project

"What would it take for the machines he made to think and talk? 'I thought I could just read the literature on how kids do it, and that would give me a blueprint for building my language and learning robots,' Roy told me."

Deb Roy

Rupal Patel

(as reported in the Guardian 2018)

The Human Speechome Project

"Over dinner one night, he boasted to Patel, who was then completing her PhD in human speech pathology, that he had already created a robot that was learning the same way kids learn. He was convinced that if it got the sort of input children get, the robot could learn from it."

Deb Roy

Rupal Patel

(as reported in the Guardian 2018)

The Human Speechome Project

"In all, they had captured 90,000 hours of video and 140,000 hours of audio. The 200 terabytes of data covered 85% of the first three years of their son's life (and 18 months of his little sister's)."

Deb Roy

Rupal Patel

(as reported in the Guardian 2018)

The Human Speechome Project

" 'He said *fah*,' Roy explained, 'but he was actually clearly referring to a fish on the wall that we were both looking at. The way I knew it was not just coincidence was that right after he looked at it and said it, he turned to me. And he had this kind of look, like a cartoon lightbulb going off – an *Ah, now I get it* kind of look. He's not even a year old, but there's a conscious being, in the sense of being self-reflective.' "

Deb Roy

Rupal Patel

(as reported in the Guardian 2018)

The Human Speechome Project

" 'I guess, putting on my AI hat, it was a humbling lesson,' he continued. 'A lesson of like, holy shit, there's a lot more here.' "

Deb Roy

Rupal Patel

(as reported in the Guardian 2018)

The Human Speechome Project

"Watching his son, Roy had been blown away by 'the incredible sophistication of what a language learner in the flesh actually looks like and does'."

Deb Roy

Rupal Patel

(as reported in the Guardian 2018)

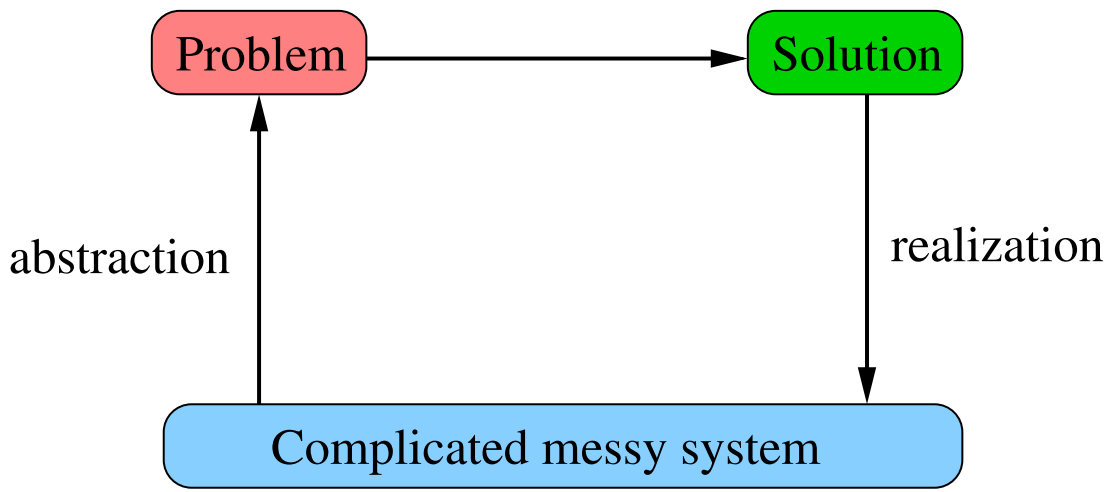
Part II

What is a learning problem?

Defining a Language-Learning Problem

1. What does it mean to know a language?
2. What does it mean to come by this language from experience?

One strategy is to identify *simple and clear* versions of these problems.



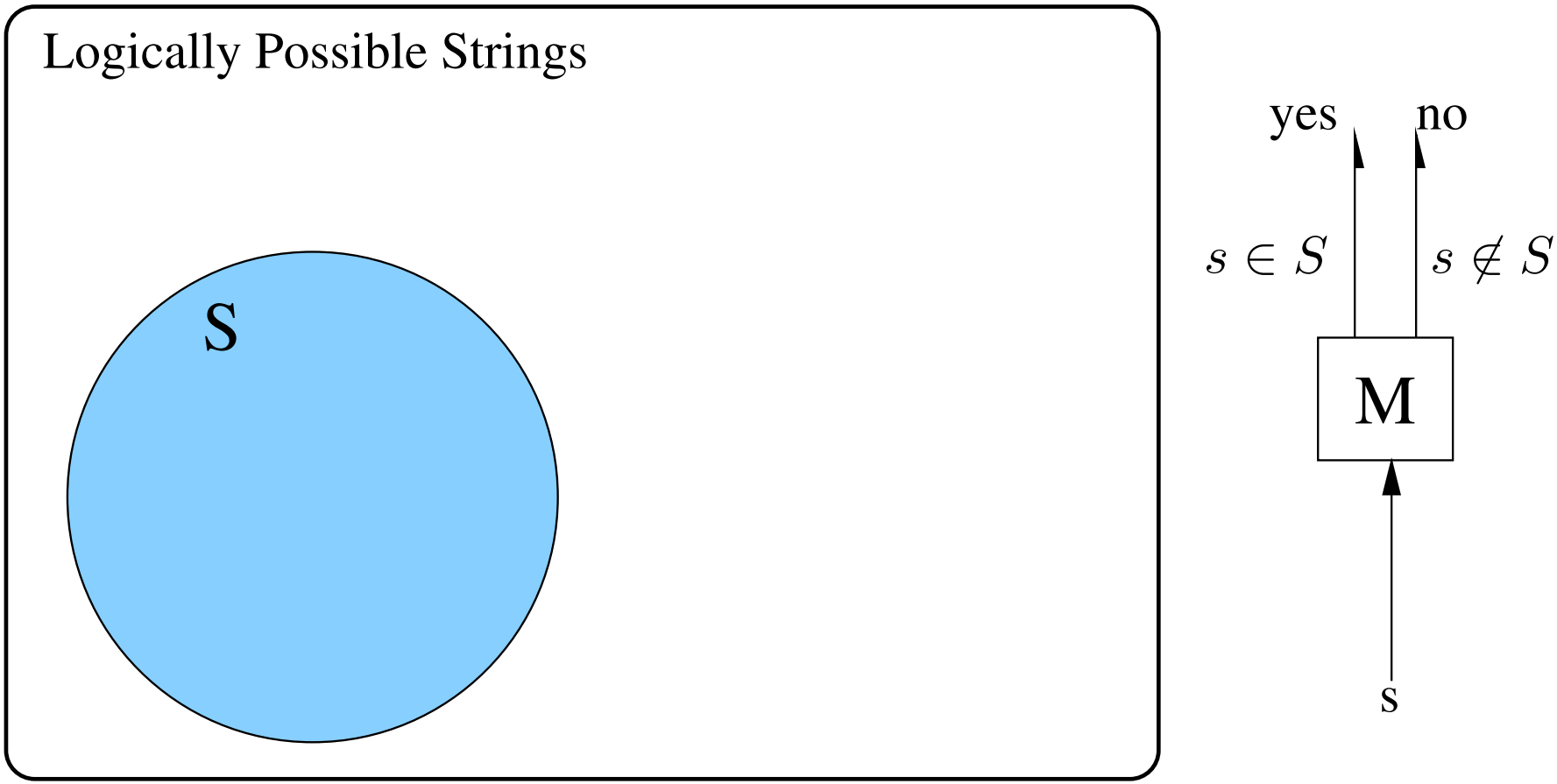
1. What does it mean to know a language?

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

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

1. What does it mean to know a language?

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
...	

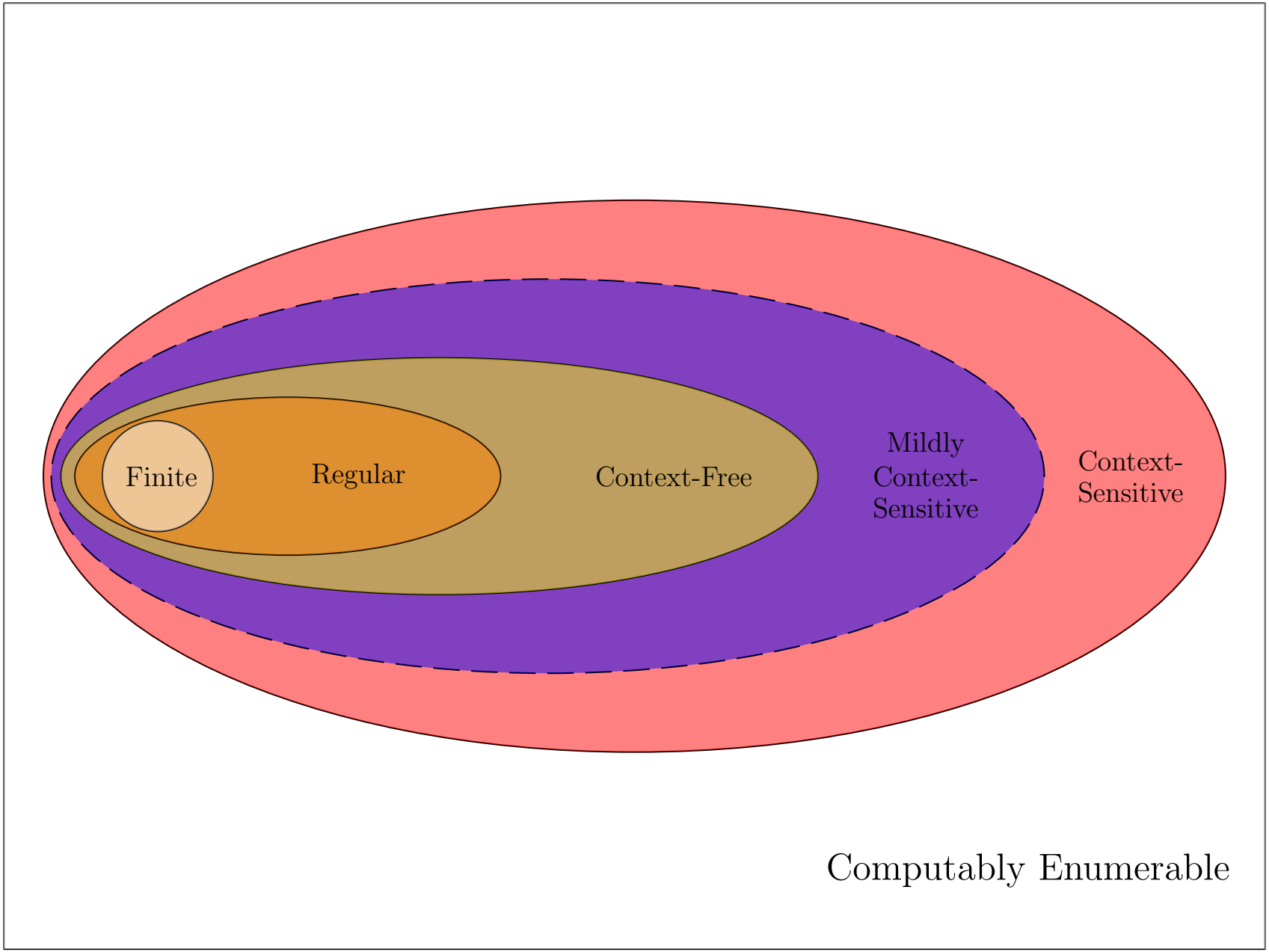
Variations thereof

Functions on the string domain ...

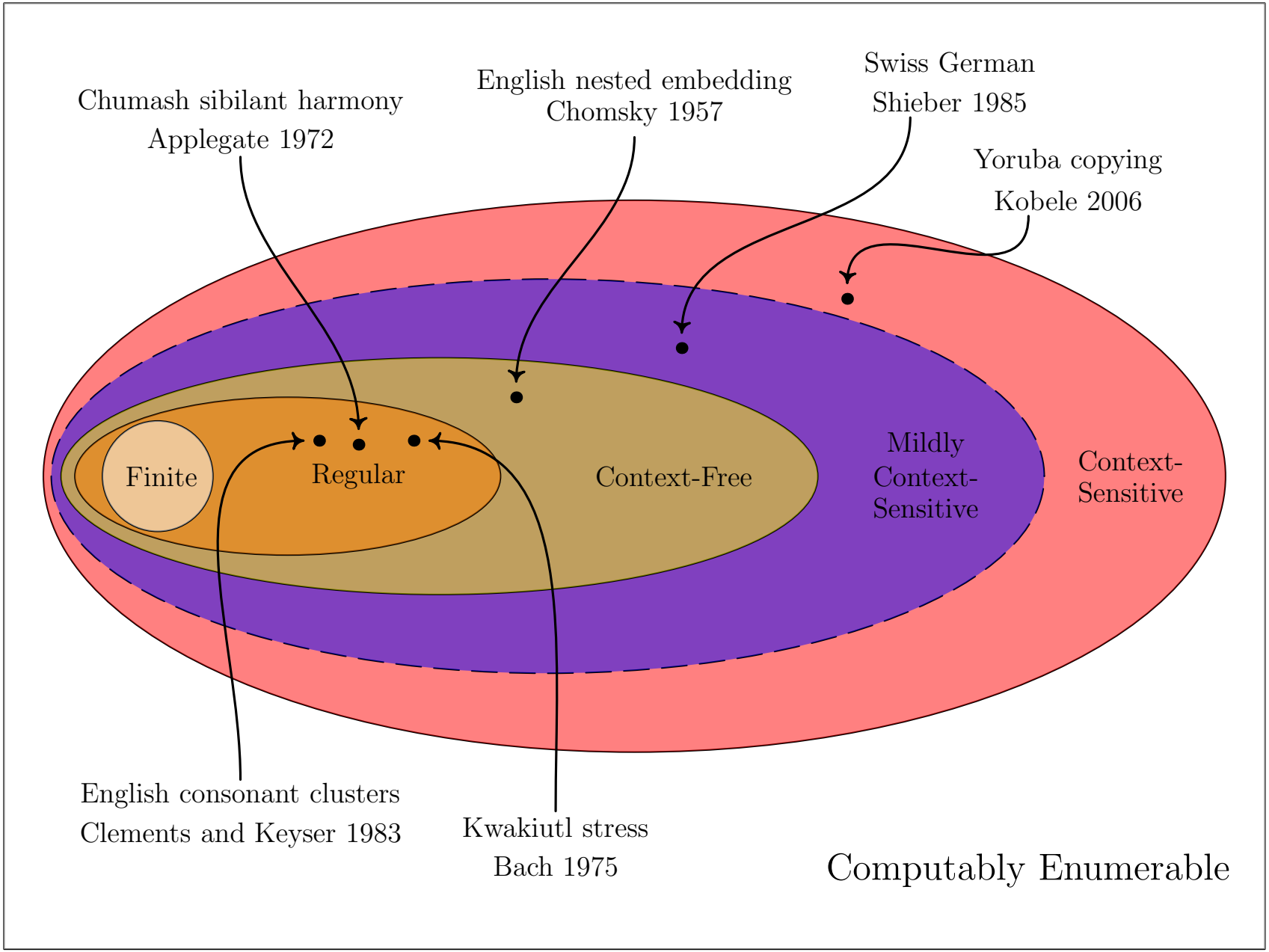
Function Type	Output Type
$\Sigma^* \to \{T, F\}$	Booleans
$\Sigma^* \to \Sigma^*$	Strings
$\Sigma^* \to \mathbb{N}$	Natural Numbers
$\Sigma^* \to [0, 1]$	Reals in the Unit Interval
$\Sigma^* \to P(\Sigma^*)$	Stringsets
...	

How are functions like those above classified?

Chomsky Hierarchy



Chomsky Hierarchy

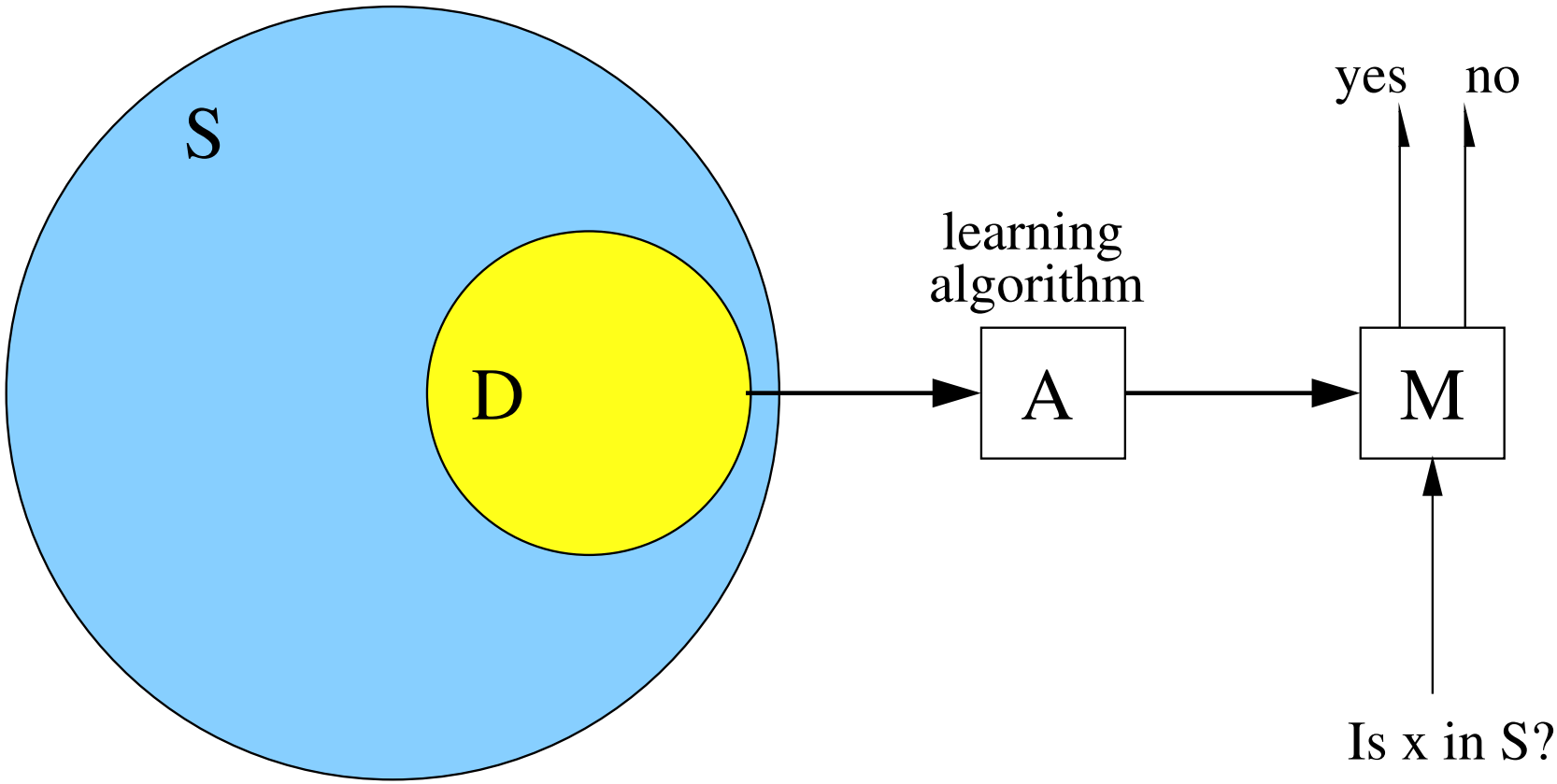


2. What does it mean to come by this knowledge from experience?

1. Which target functions?
2. What kinds of experience?
3. What counts as success?

Answering questions 1-2 are important for defining the **instance space** of the learning problem. Question 3 is about conditions on successful solutions.

In Pictures



for any S belonging to a class C and for any D from some 'legitimate' class of experience?

NOT ABOUT METHODS (MANY, MANY METHODS)

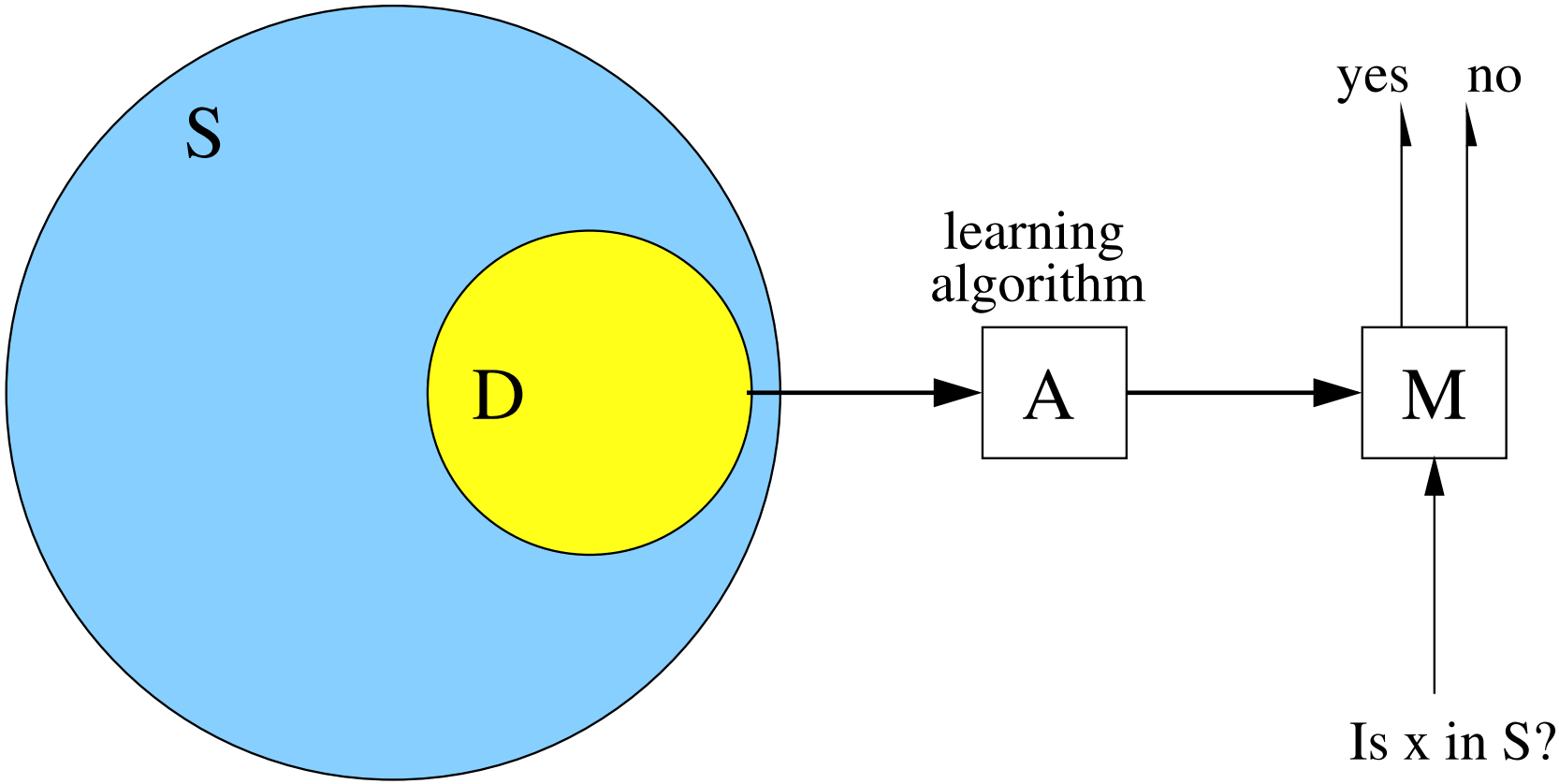
1. Connectionism/Associative Learning (Rosenblatt 1959, McClelland and Rumelhart 1986, Kapatsinski 2018, a.o.)
2. Bayesian methods (Bishop 2006, Kemp and Tenenbaum 2008, a.o.)
3. Probabilistic Graphical Models (Pearl 1988, Koller and Friedman 2010, a.o.)
4. State-merging (Feldman 1972, Angluin 1982, 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. Support Vector Machines (Vapnik 1995, 1998 a.o.)
8. ...

NOT about methods (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. ...

In Pictures



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

Computational Learning Theory

1. Identifications in the Limit (Gold 1967)
2. Identifications in the Limit with probability p with stochastic input (Angluin 1988a)
3. Active/Query Learning (Angluin 1988b)
4. Probably Approximately Correct (PAC) Learning (Valiant 1984)
5. Optimizing Objective Functions
6. ...

Filling out the Instance Space: The Experience

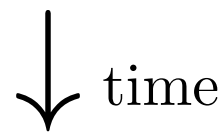
1. It is a sequence.
2. It is finite.

$$
\begin{array}{l}
w_0 \\
w_1 \\
w_2 \\
\ldots \\
w_n
\end{array}
\qquad \downarrow \text{time}
$$

Filling out the Instance Space: Types of Experience

1. Positive evidence

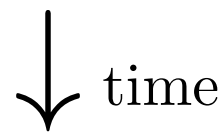
$$
\begin{aligned}
w_0 &\in L \\
w_1 &\in L \\
w_2 &\in L \\
&\dots \\
w_n &\in L
\end{aligned}
$$



Filling out the Instance Space: Types of Experience

2. Positive and negative evidence

$$
\begin{aligned}
w_0 &\in L \\
w_1 &\notin L \\
w_2 &\notin L \\
&\dots \\
w_n &\in L
\end{aligned}
$$



Filling out the Instance Space: Types of Experience

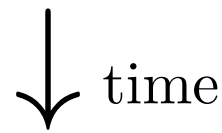
3. Noisy evidence

$$
\begin{array}{c}
w_0 \in L \\
w_1 \notin L \\
w_2 \in L \text{ (\textbf{but in fact} } w_2 \notin L) \\
\dots \\
w_n \in L
\end{array}
\qquad \downarrow \text{ time}
$$

Filling out the Instance Space: Types of Experience

4. Queried Evidence

$$w_0 \in L$$
$$w_1 \notin L$$
$$w_2 \in L \text{ (because learner specifically asked about } w_2)$$
$$\ldots$$
$$w_n \in L$$



What counts as success?

1. Convergence.
2. Imagine an infinite sequence. Is there some point n after which the learner's hypothesis doesn't change (much)?

datum	Learner's Hypothesis
w_0	$\varphi(\langle w_0 \rangle) = G_0$

What counts as success?

1. Convergence.
2. Imagine an infinite sequence. Is there some point n after which the learner's hypothesis doesn't change (much)?

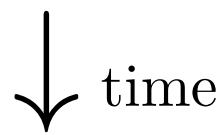
datum	Learner's Hypothesis
w_0	$\varphi(\langle w_0 \rangle) = G_0$
w_1	$\varphi(\langle w_0, w_1 \rangle) = G_1$

↓ time

What counts as success?

1. Convergence.
2. Imagine an infinite sequence. Is there some point n after which the learner's hypothesis doesn't change (much)?

datum	Learner's Hypothesis
w_0	$\varphi(\langle w_0 \rangle) = G_0$
w_1	$\varphi(\langle w_0, w_1 \rangle) = G_1$
w_2	$\varphi(\langle w_0, w_1, w_2 \rangle) = G_2$



What counts as success?

1. Convergence.
2. Imagine an infinite sequence. Is there some point n after which the learner's hypothesis doesn't change (much)?

datum	Learner's Hypothesis
w_0	$\varphi(\langle w_0 \rangle) = G_0$
w_1	$\varphi(\langle w_0, w_1 \rangle) = G_1$
w_2	$\varphi(\langle w_0, w_1, w_2 \rangle) = G_2$
$\dots$	

What counts as success?

1. Convergence.
2. Imagine an infinite sequence. Is there some point n after which the learner's hypothesis doesn't change (much)?

datum	Learner's Hypothesis
w_0	$\varphi(\langle w_0 \rangle) = G_0$
w_1	$\varphi(\langle w_0, w_1 \rangle) = G_1$
w_2	$\varphi(\langle w_0, w_1, w_2 \rangle) = G_2$
$\ldots$	
w_n	$\varphi(\langle w_0, w_1, w_2, \ldots, w_n \rangle) = G_n$

What counts as success?

1. Convergence.
2. Imagine an infinite sequence. Is there some point n after which the learner's hypothesis doesn't change (much)?

datum	Learner's Hypothesis
w_0	$\varphi(\langle w_0 \rangle) = G_0$
w_1	$\varphi(\langle w_0, w_1 \rangle) = G_1$
w_2	$\varphi(\langle w_0, w_1, w_2 \rangle) = G_2$
$\ldots$	
w_n	$\varphi(\langle w_0, w_1, w_2, \ldots, w_n \rangle) = G_n$
$\ldots$	

What counts as success?

1. Convergence.
2. Imagine an infinite sequence. Is there some point n after which the learner's hypothesis doesn't change (much)?

datum	Learner's Hypothesis
w_0	$\varphi(\langle w_0 \rangle) = G_0$
w_1	$\varphi(\langle w_0, w_1 \rangle) = G_1$
w_2	$\varphi(\langle w_0, w_1, w_2 \rangle) = G_2$
$\ldots$	
w_n	$\varphi(\langle w_0, w_1, w_2, \ldots, w_n \rangle) = G_n$
$\ldots$	
w_m	$\varphi(\langle w_0, w_1, w_2, \ldots, w_m \rangle) = G_m$

$\downarrow$ time

Does
$G_m \simeq G_n$**?**

Filling out the Instance Space: Which experiences?

Types of Experience

1. Positive-only or positive and negative evidence.
2. Noiseless or noisy evidence.
3. Queries allowed or not?

Which infinite sequences require convergence?

1. only **complete** ones? I.e. where every piece of information occurs at some finite point
2. only **computable** ones? I.e. the infinite sequence itself is describable by some grammar

Some Parameters for Defining Learning Problems

Makes learning easier	**Makes learning harder**
positive and negative evidence	positive evidence only
noiseless evidence	noisy evidence
queries permitted	queries not permitted
approximate convergence	exact convergence
complete infinite sequences	any infinite sequence
computable infinite sequences	any infinite sequence

Some Parameters for Defining Learning Problems

Makes learning easier	Makes learning harder
positive and negative evidence	positive evidence only
noiseless evidence	noisy evidence
queries permitted	queries not permitted
approximate convergence	exact convergence
complete infinite sequences	any infinite sequence
computable infinite sequences	any infinite sequence

1. Identification in the limit from positive data (Gold 1967)

Some Parameters for Defining Learning Problems

Makes learning easier	Makes learning harder
positive and negative evidence	positive evidence only
noiseless evidence	noisy evidence
queries permitted	queries not permitted
approximate convergence	exact convergence
complete infinite sequences	any infinite sequence
computable infinite sequences	any infinite sequence

2. Identification in the limit from positive and negative data
(Gold 1967)

Some Parameters for Defining Learning Problems

Makes learning easier	Makes learning harder
positive and negative evidence	positive evidence only
noiseless evidence	noisy evidence
queries permitted	queries not permitted
approximate convergence	exact convergence
complete infinite sequences	any infinite sequence
computable infinite sequences	any infinite sequence

3. Identification in the limit from positive data from c.e. texts
   (Gold 1967)

4. Learning context-free and c.e. distributions
   (Horning 1969, Angluin 1988)

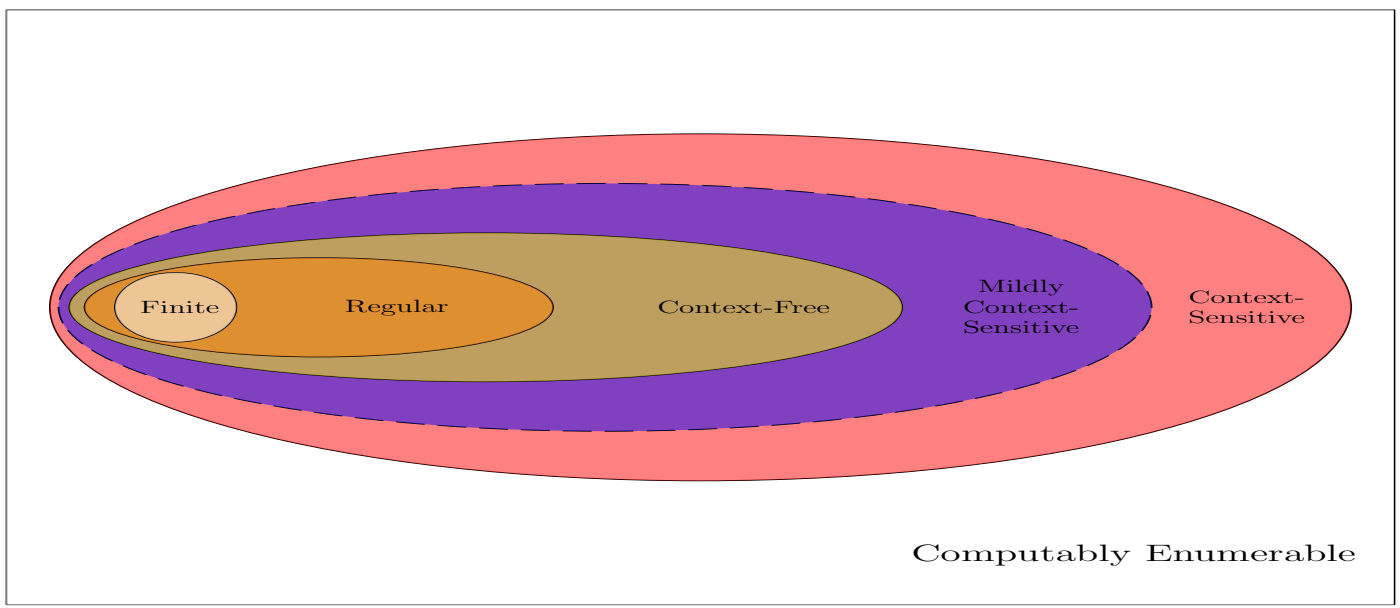
Some Parameters for Defining Learning Problems

Makes learning easier	Makes learning harder
positive and negative evidence	positive evidence only
noiseless evidence	noisy evidence
queries permitted	queries not permitted
approximate convergence	exact convergence
complete infinite sequences	any infinite sequence
computable infinite sequences	any infinite sequence

5. Probably Approximately Correct learning
   (Valiant 1984, Anthony and Biggs 1991, Kearns and Vazirani 1994

Results of Formal Learning Theories: Existence

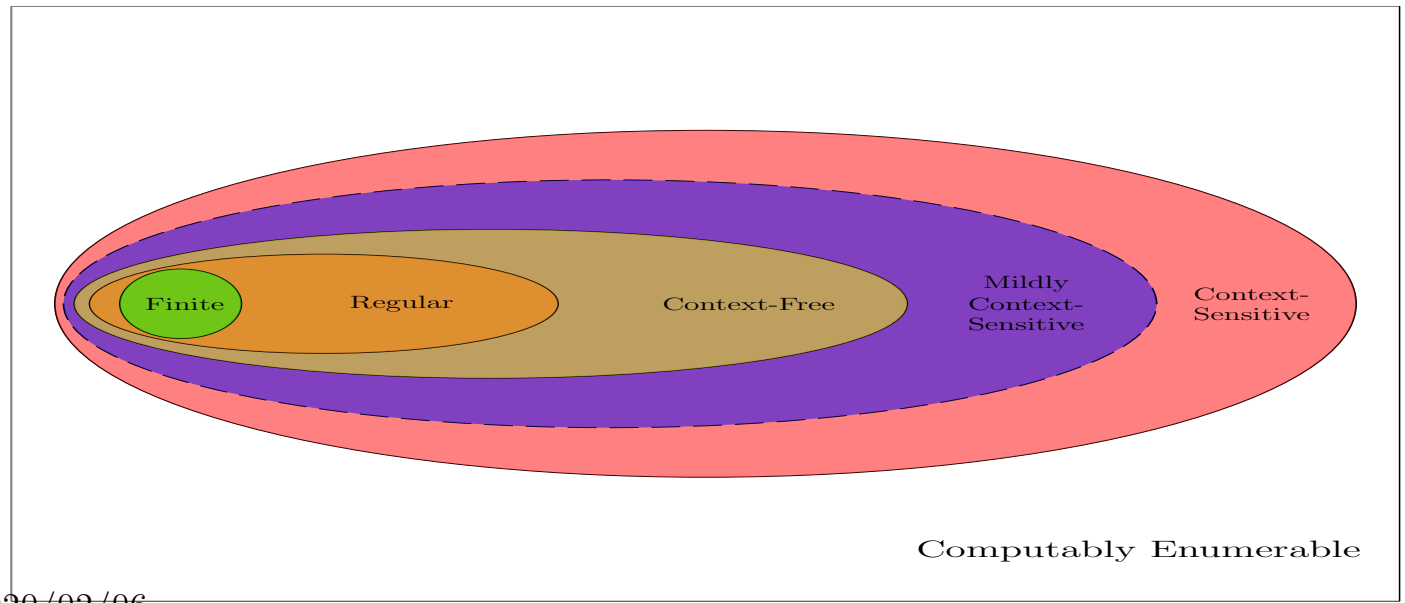
Makes learning easier	Makes learning harder
positive and negative evidence	positive evidence only
noiseless evidence	noisy evidence
queries permitted	queries not permitted
approximate convergence	exact convergence
complete infinite sequences	any infinite sequence
computable infinite sequences	any infinite sequence



Results of Formal Learning Theories: Existence

Makes learning easier	Makes learning harder
positive and negative evidence	positive evidence only
noiseless evidence	noisy evidence
queries permitted	queries not permitted
approximate convergence	exact convergence
complete infinite sequences	any infinite sequence
computable infinite sequences	any infinite sequence

1. Identification in the limit from positive data (Gold 1967)

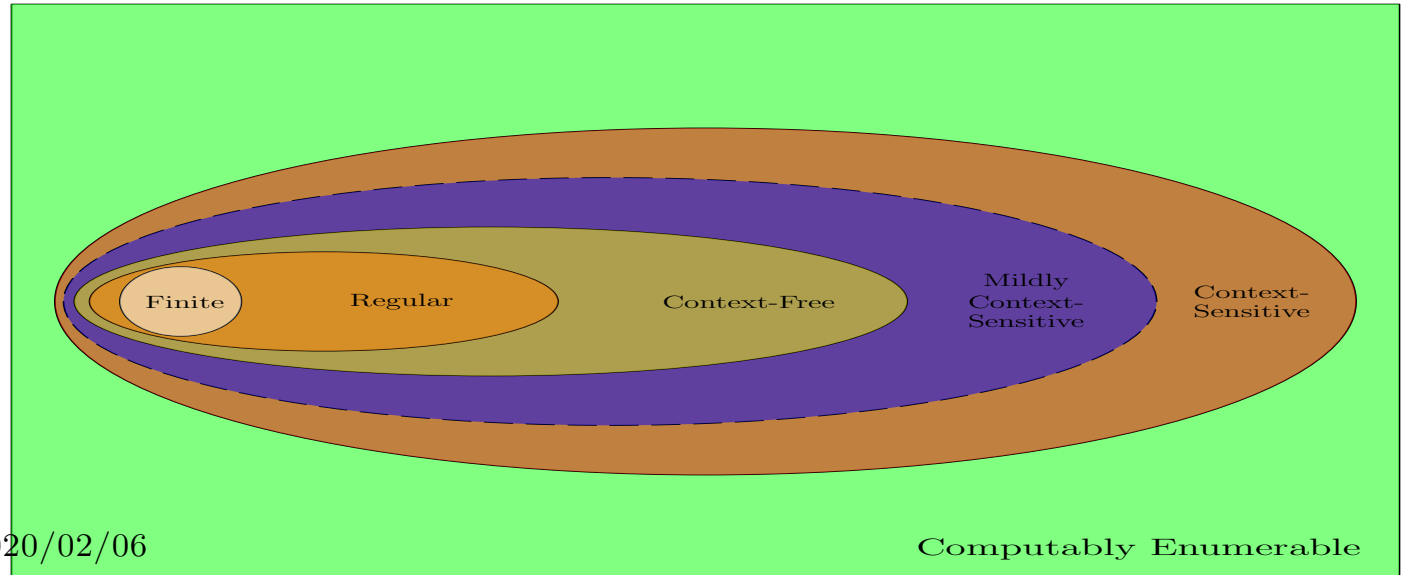


Results of Formal Learning Theories: Existence

Makes learning easier	Makes learning harder
positive and negative evidence	positive evidence only
noiseless evidence	noisy evidence
queries permitted	queries not permitted
approximate convergence	exact convergence
complete infinite sequences	any infinite sequence
computable infinite sequences	any infinite sequence

2. Identification in the limit from positive and negative data

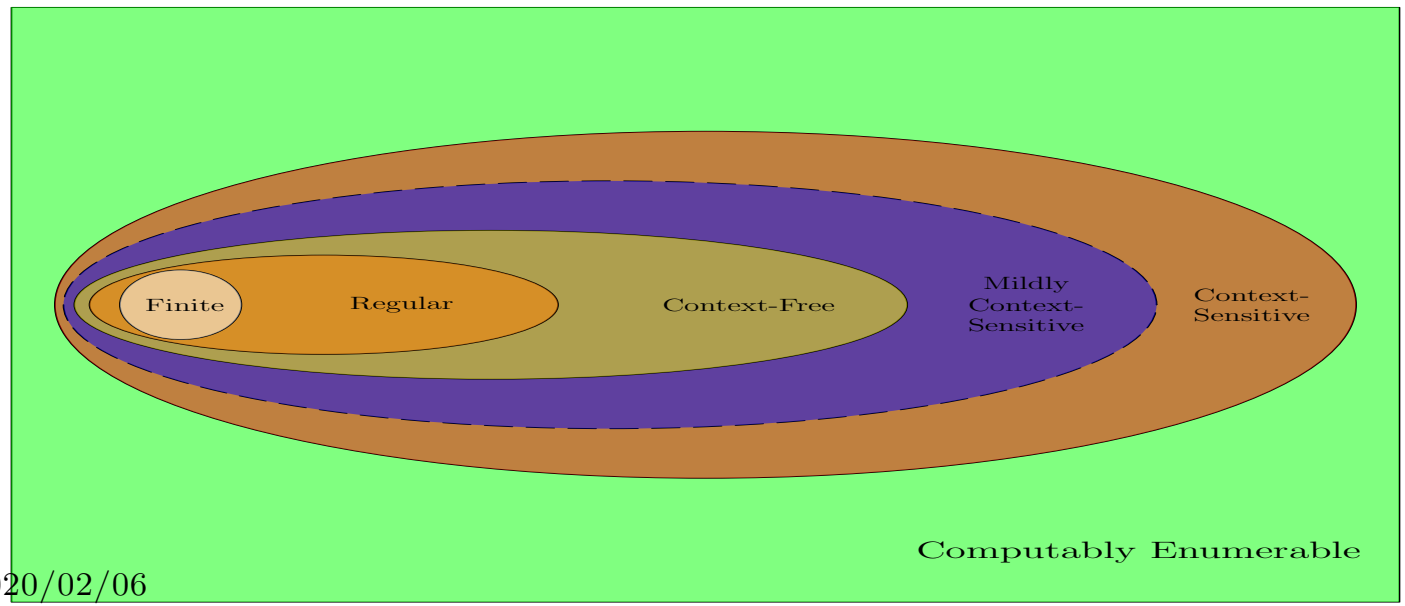
(Gold 1967)



Results of Formal Learning Theories: Existence

Makes learning easier	Makes learning harder
positive and negative evidence	positive evidence only
noiseless evidence	noisy evidence
queries permitted	queries not permitted
approximate convergence	exact convergence
complete infinite sequences	any infinite sequence
computable infinite sequences	any infinite sequence

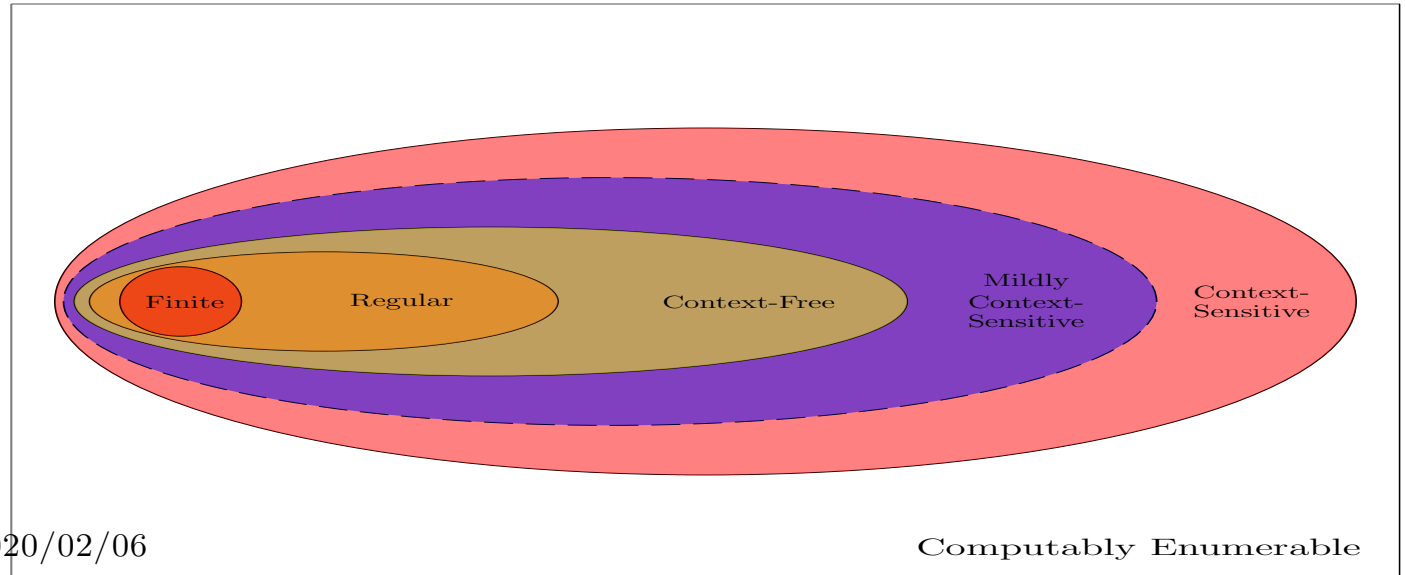
3. Identification in the limit from positive data from c.e. texts (Gold 1967)
4. Learning context-free and c.e. distributions (Horning 1969, Angluin 1988)



Results of Formal Learning Theories: Existence

Makes learning easier	Makes learning harder
positive and negative evidence	positive evidence only
noiseless evidence	noisy evidence
queries permitted	queries not permitted
approximate convergence	exact convergence
complete infinite sequences	any infinite sequence
computable infinite sequences	any infinite sequence

5. Probably Approximately Correct learning
   (Valiant 1984, Anthony and Biggs 1991, Kearns and Vazirani 1994



Results of Formal Learning Theory: Feasibility

Makes learning easier	**Makes learning harder**
positive and negative evidence	positive evidence only
noiseless evidence	noisy evidence
queries permitted	queries not permitted
approximate convergence	exact convergence
complete infinite sequences	any infinite sequence
computable infinite sequences	any infinite sequence

Results of Formal Learning Theory: Feasibility

Makes learning easier	Makes learning harder
positive and negative evidence	positive evidence only
noiseless evidence	noisy evidence
queries permitted	queries not permitted
approximate convergence	exact convergence
complete infinite sequences	any infinite sequence
computable infinite sequences	any infinite sequence

1. Identification in the limit from positive data (Gold 1967)

   No superfinite class is learnable.
   The finite class is feasibly learnable.

Results of Formal Learning Theory: Feasibility

Makes learning easier	Makes learning harder
positive and negative evidence	positive evidence only
noiseless evidence	noisy evidence
queries permitted	queries not permitted
approximate convergence	exact convergence
complete infinite sequences	any infinite sequence
computable infinite sequences	any infinite sequence

2. Identification in the limit from positive and negative data (Gold 1967)

   The c.e. class is learnable but NOT even the regular class is feasibly learnable (see appendix).

Results of Formal Learning Theory: Feasibility

Makes learning easier	Makes learning harder
positive and negative evidence	positive evidence only
noiseless evidence	noisy evidence
queries permitted	queries not permitted
approximate convergence	exact convergence
complete infinite sequences	any infinite sequence
computable infinite sequences	any infinite sequence

3. Identification in the limit from positive data from c.e. texts (Gold 1967)

4. Learning context-free and c.e. distributions (Horning 1969, Angluin 1988)

   The c.e. class of languages and distributions is learnable but NOT even the regular class is feasibly learnable.

Results of Formal Learning Theory: Feasibility

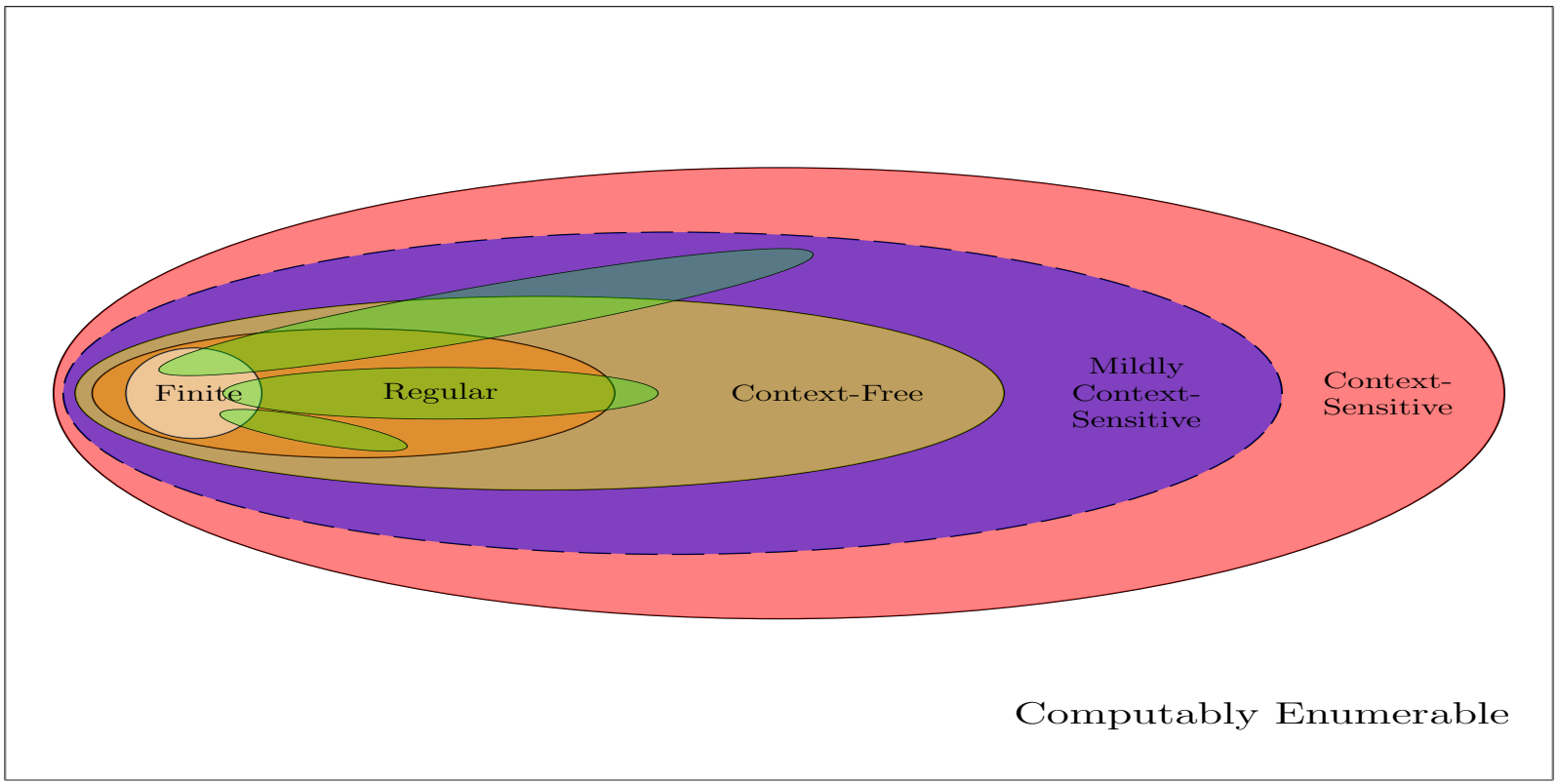
Makes learning easier	Makes learning harder
positive and negative evidence	positive evidence only
noiseless evidence	noisy evidence
queries permitted	queries not permitted
approximate convergence	exact convergence
complete infinite sequences	any infinite sequence
computable infinite sequences	any infinite sequence

5. Probably Approximately Correct learning
   (Valiant 1984, Anthony and Biggs 1991, Kearns and Vazirani 1994)

   Not even the finite class of languages is learnable.

Formal Learning Theory: Positive Results

Many classes which cross-cut the Chomsky hierarchy and exclude some finite languages are feasibly learnable in the senses discussed.



(Angluin 1980, 1982, Garcia et al. 1990, Muggleton 1990, Denis et al. 2002, Fernau 2003, Yokomori 2003, Oates et al. 2006, Niyogi 2006, Clark and Eryaud 2007, Heinz 2008, to appear, Yoshinaka 2008, Case et al. 2009, de la Higuera 2010)

Summary

1. Structured, restricted hypothesis spaces can be feasibly learned.
2. The positive learning results are proven results, and the proofs are often constructive.
3. The claim that "statistical learning" is more powerful than "symbolic learning" mischaracterizes the learning issues.
4. The real issue is how to reduce the size of the instance space.

Gold 1967

Gold provides three ways to interpret his three main results:

1. "The class of natural languages is much smaller than one would expect from our present models of syntax. That is, even if English is context-sensitive, it is not true that any context-sensitive language can occur naturally . . ."
2. "The child receives negative instances by being corrected in a way that we do not recognize . . ."
3. "There is an a priori restriction on the class of texts [presentations of data; i.e. infinite sequences of experience] which can occur . . ."

(Gold 1967: 453-4)

Part IV

Consequences of CLT
(or The Perceptual Misconception Machine)

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.

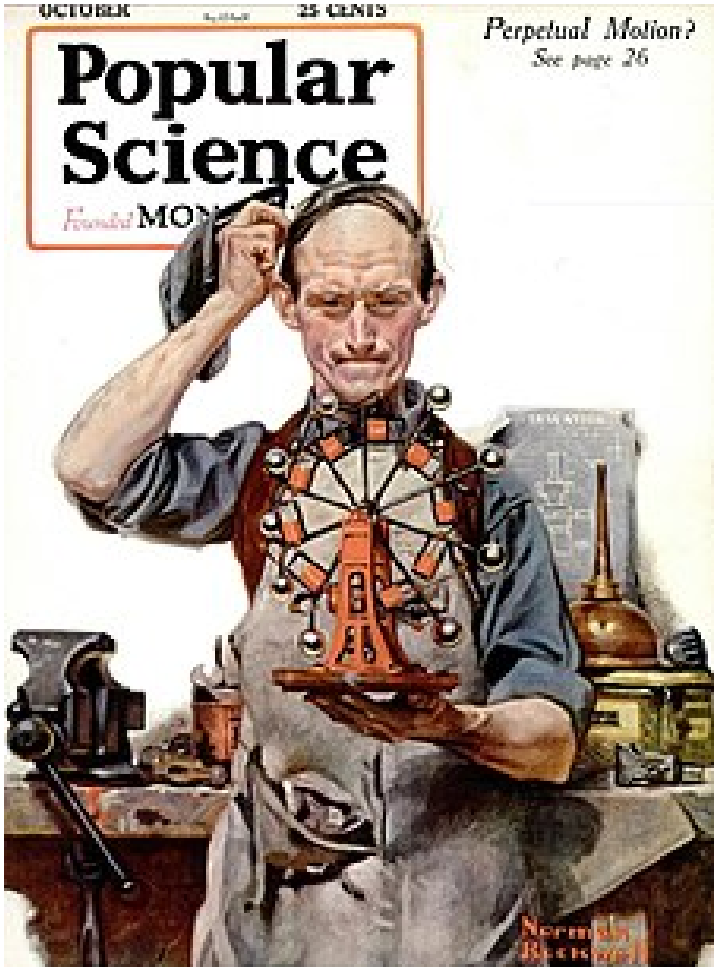
The main lesson from CLT

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.

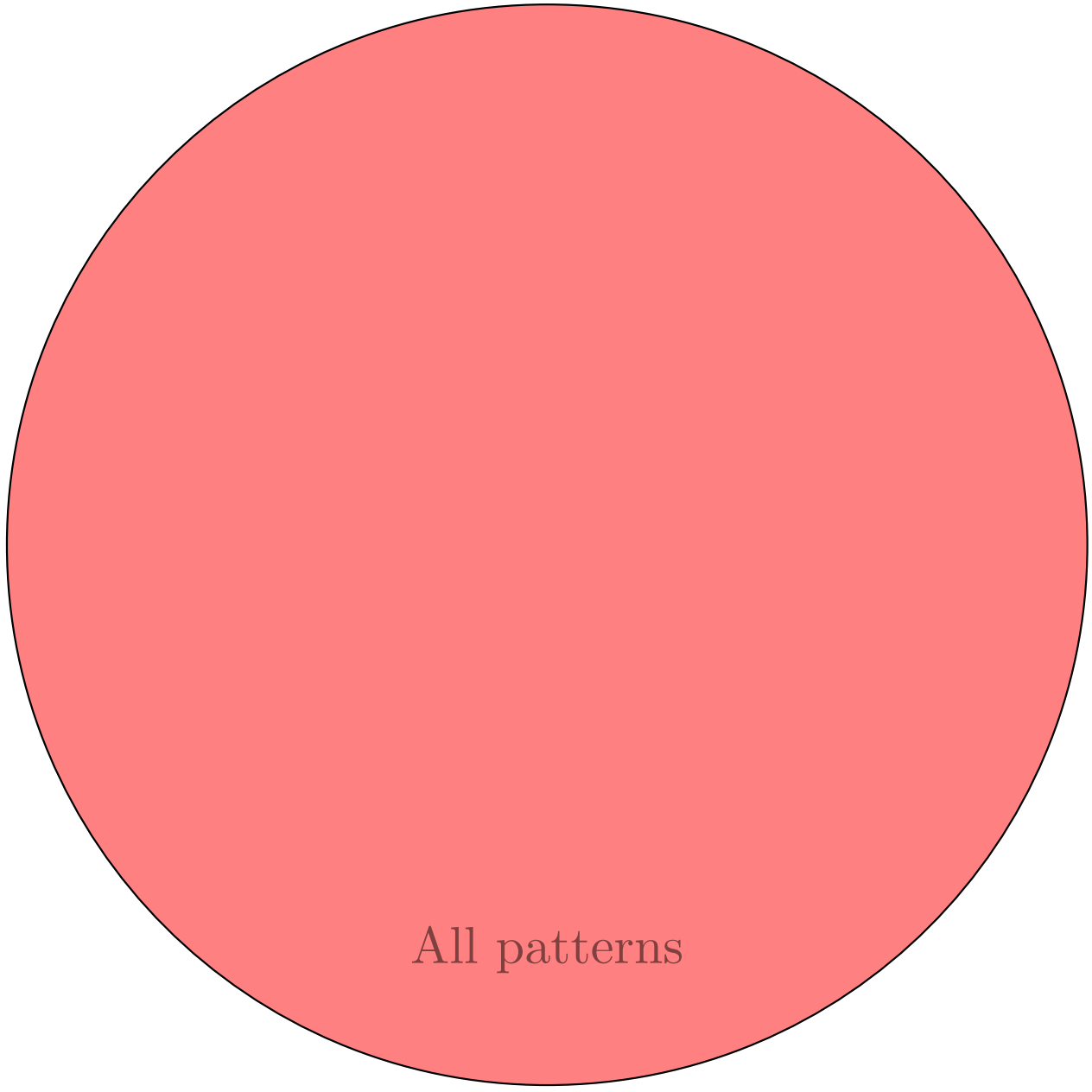
1. The class C cannot be all patterns, or even all computable patterns.
2. Class C must have *more structure*, and many logically possible patterns *must be outside* of C.
3. There is no avoiding prior knowledge.

Computational Laws of Learning

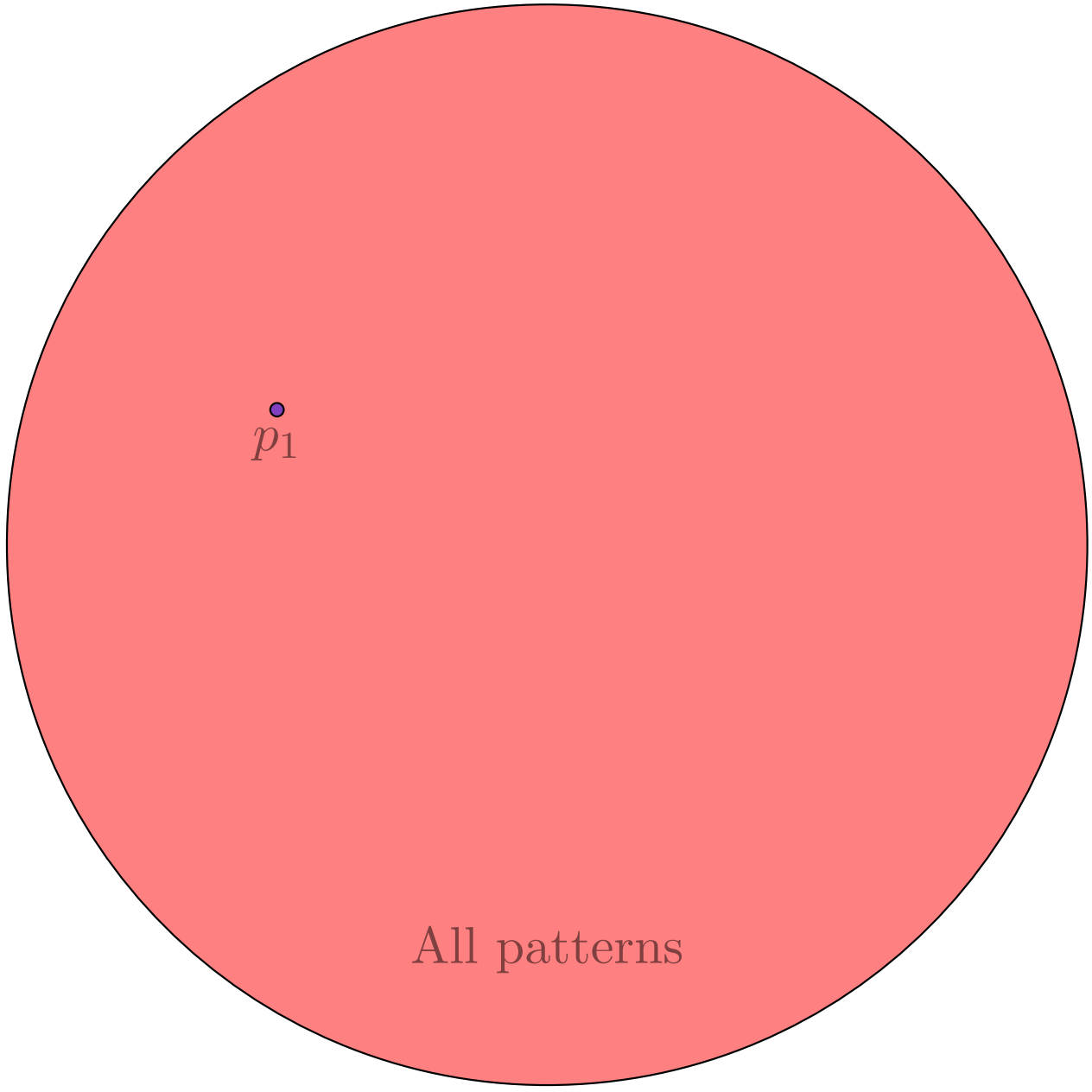
Feasibly solving a learning problem requires defining a target class C of patterns.

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)

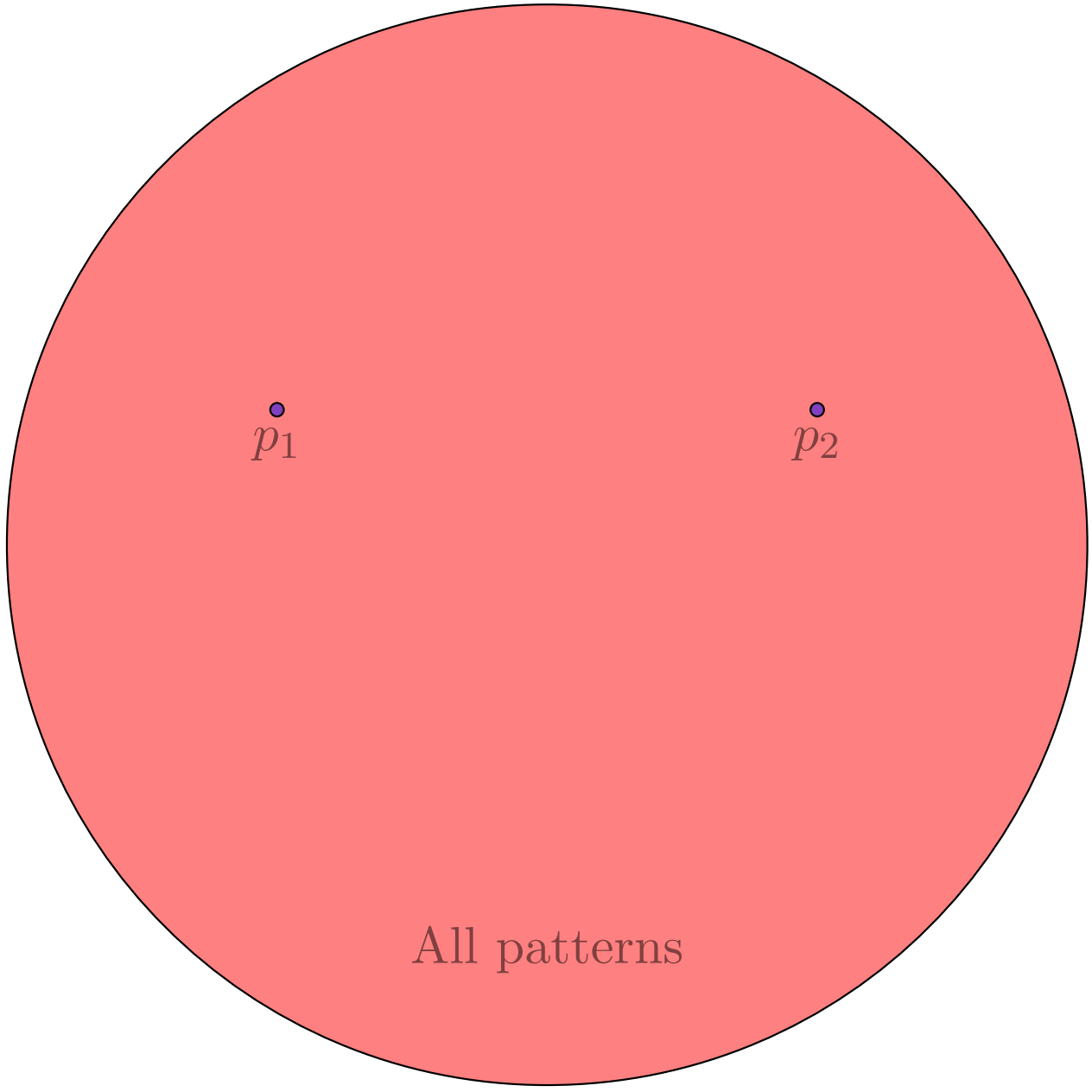
In Pictures: Given ML algorithm A



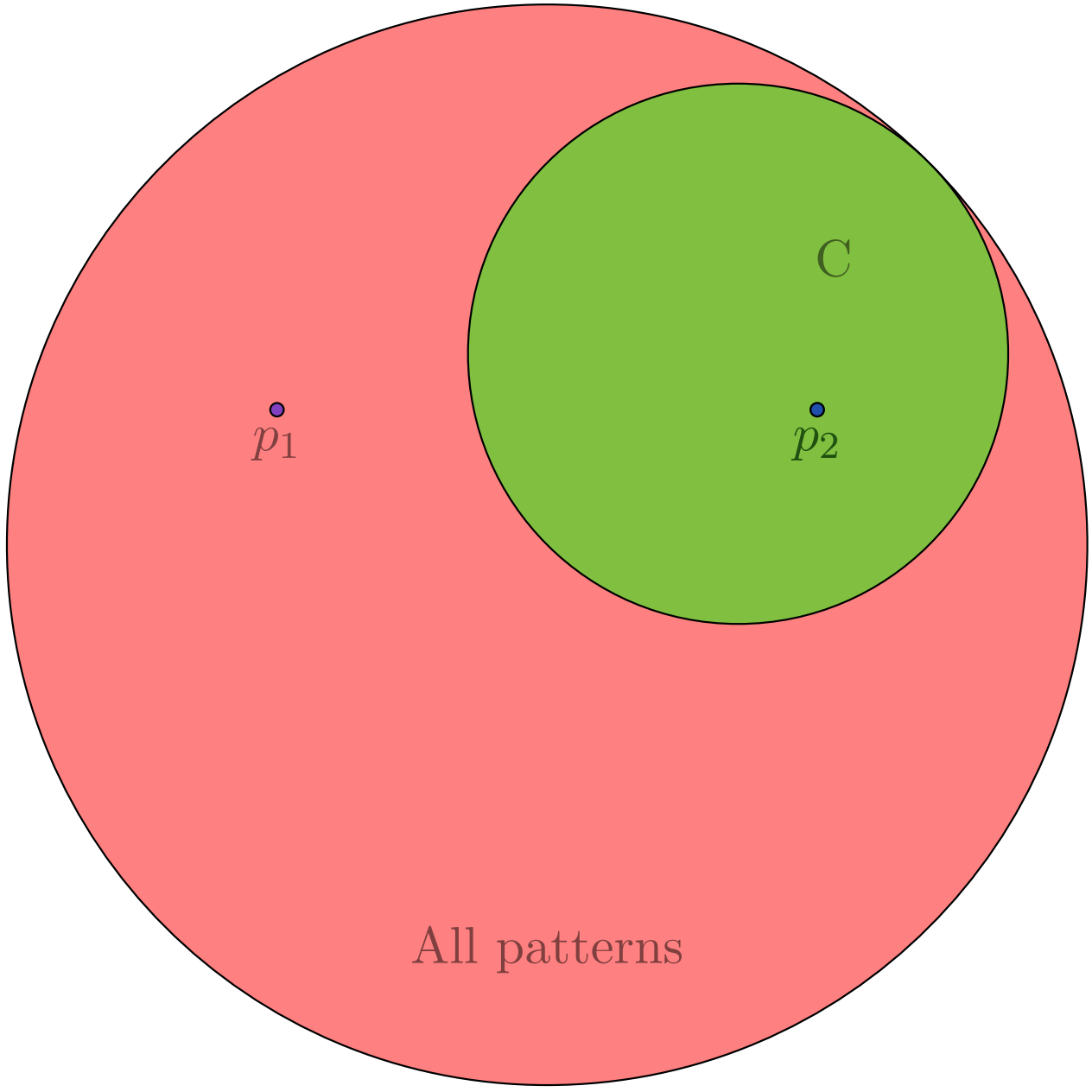
In Pictures: Given ML algorithm A



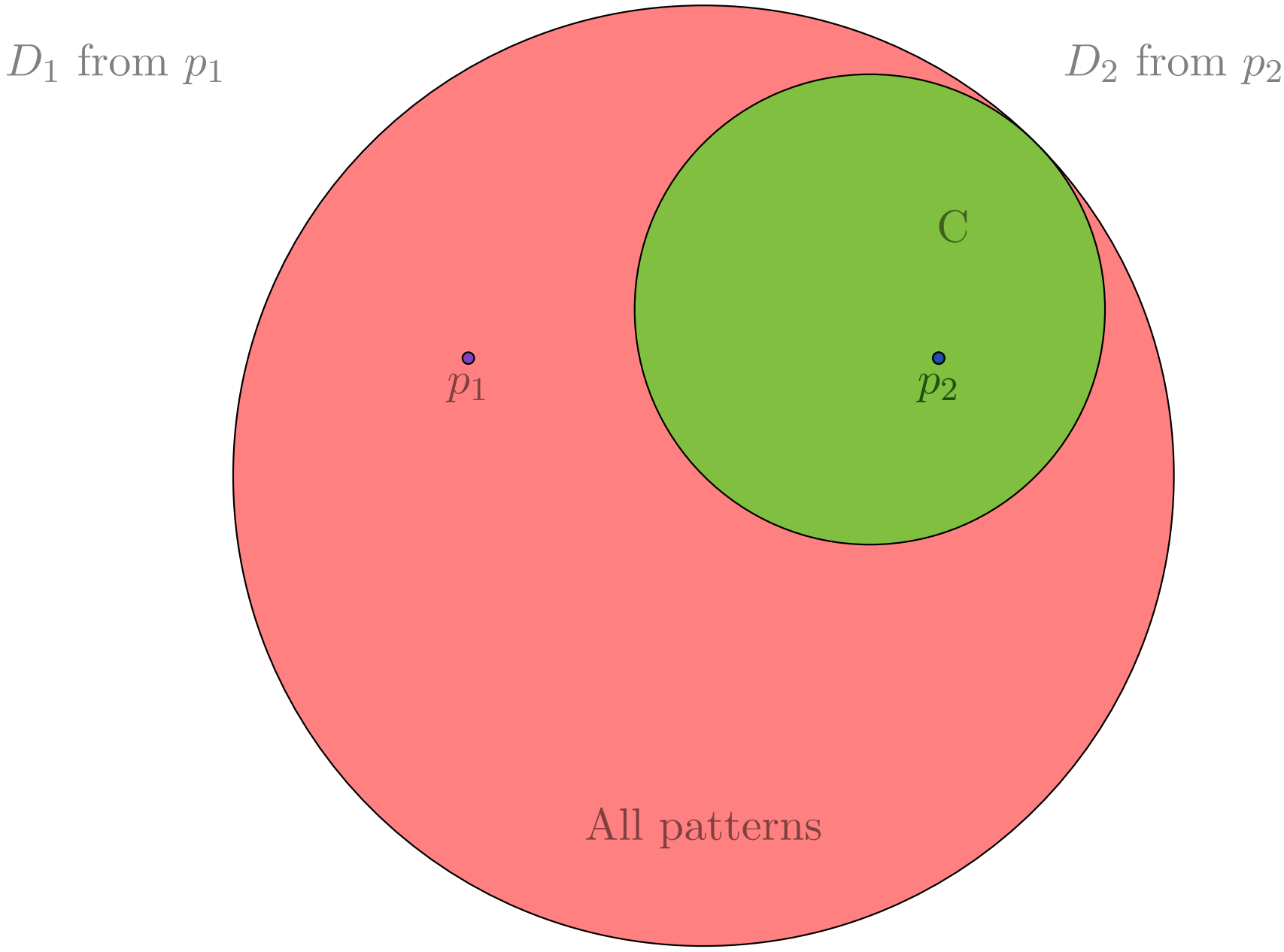
In Pictures: Given ML algorithm A



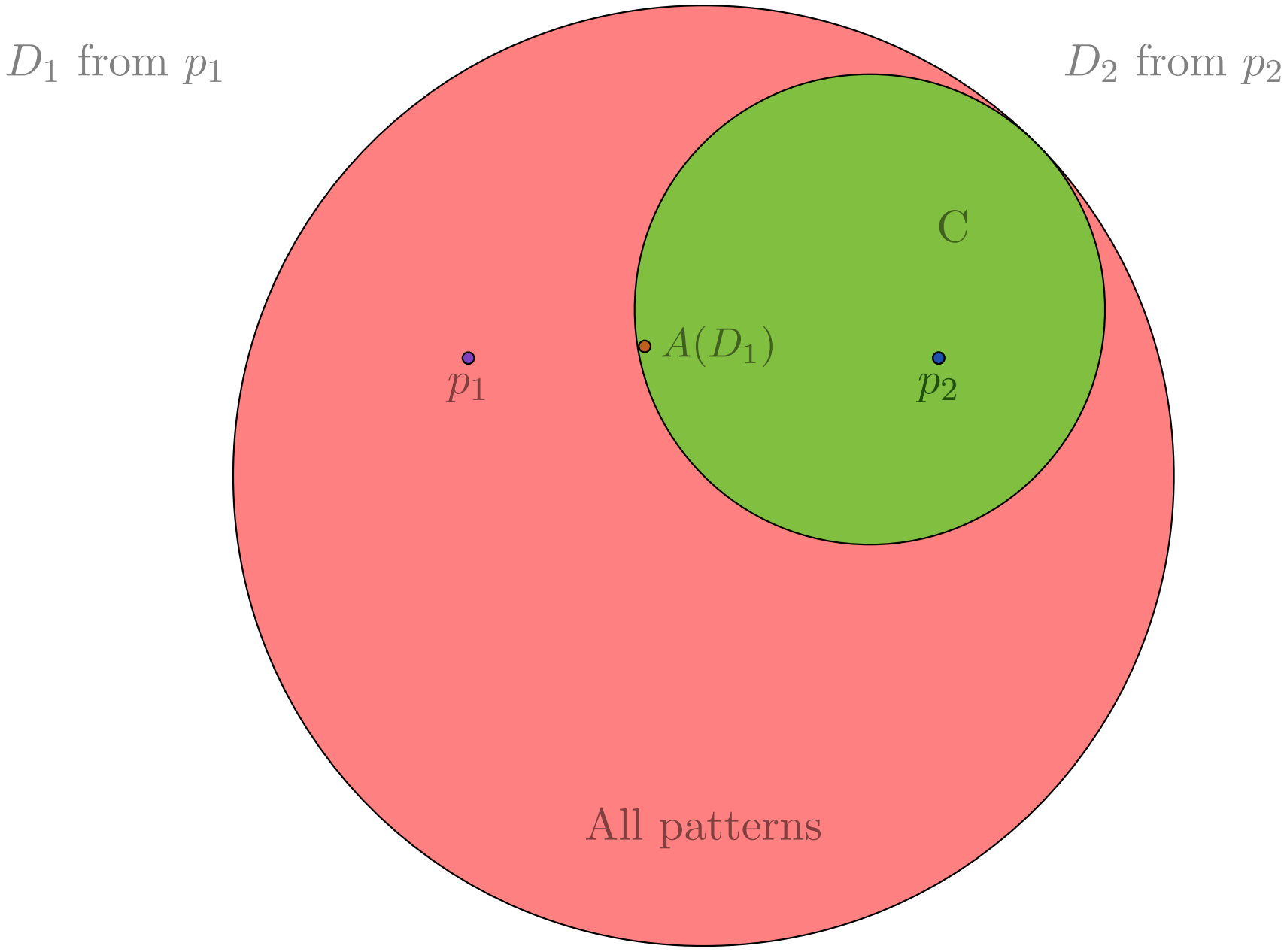
In Pictures: Given ML algorithm A



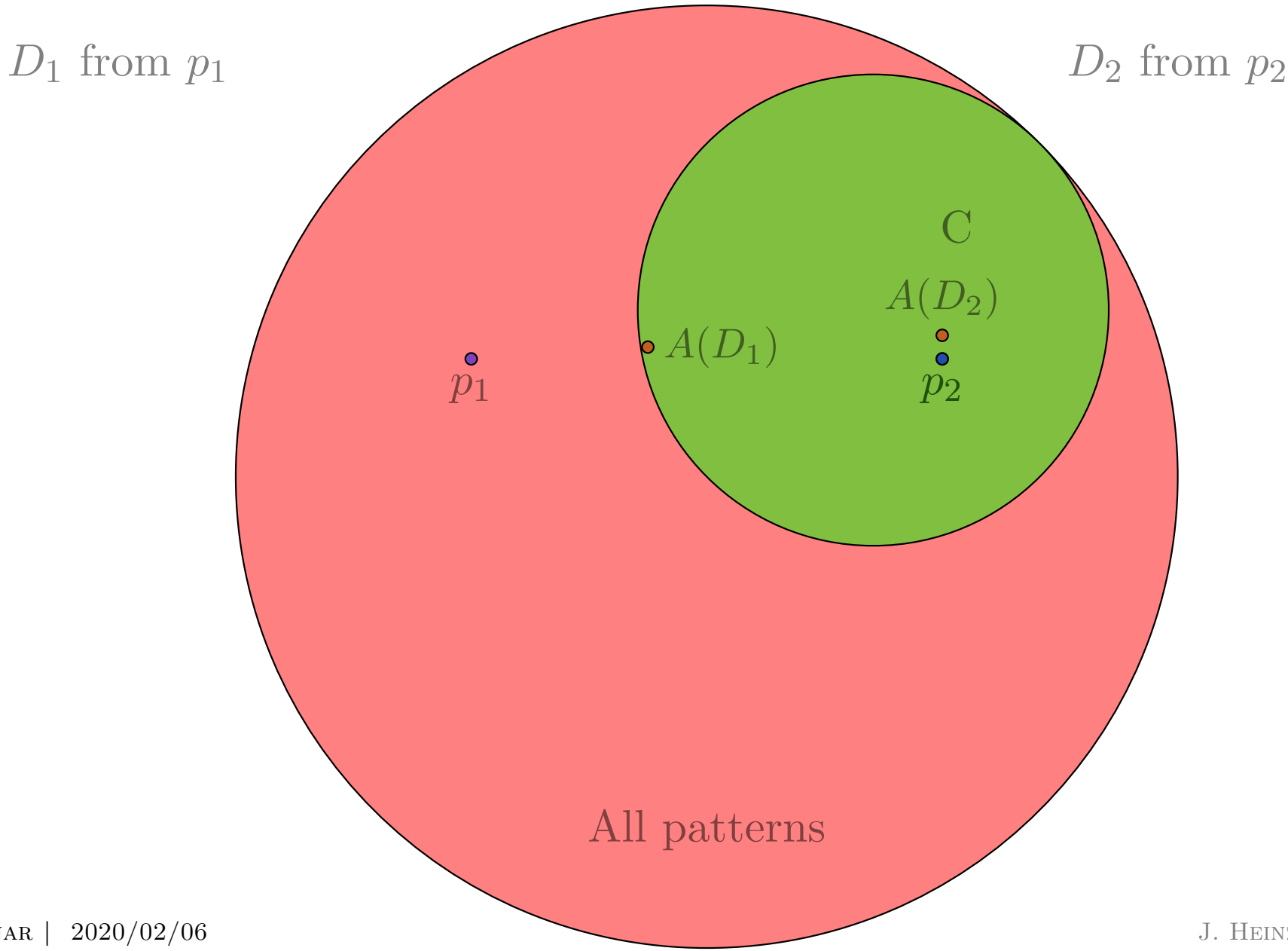
In Pictures: Given ML algorithm A



In Pictures: Given ML algorithm A



In Pictures: Given ML algorithm A

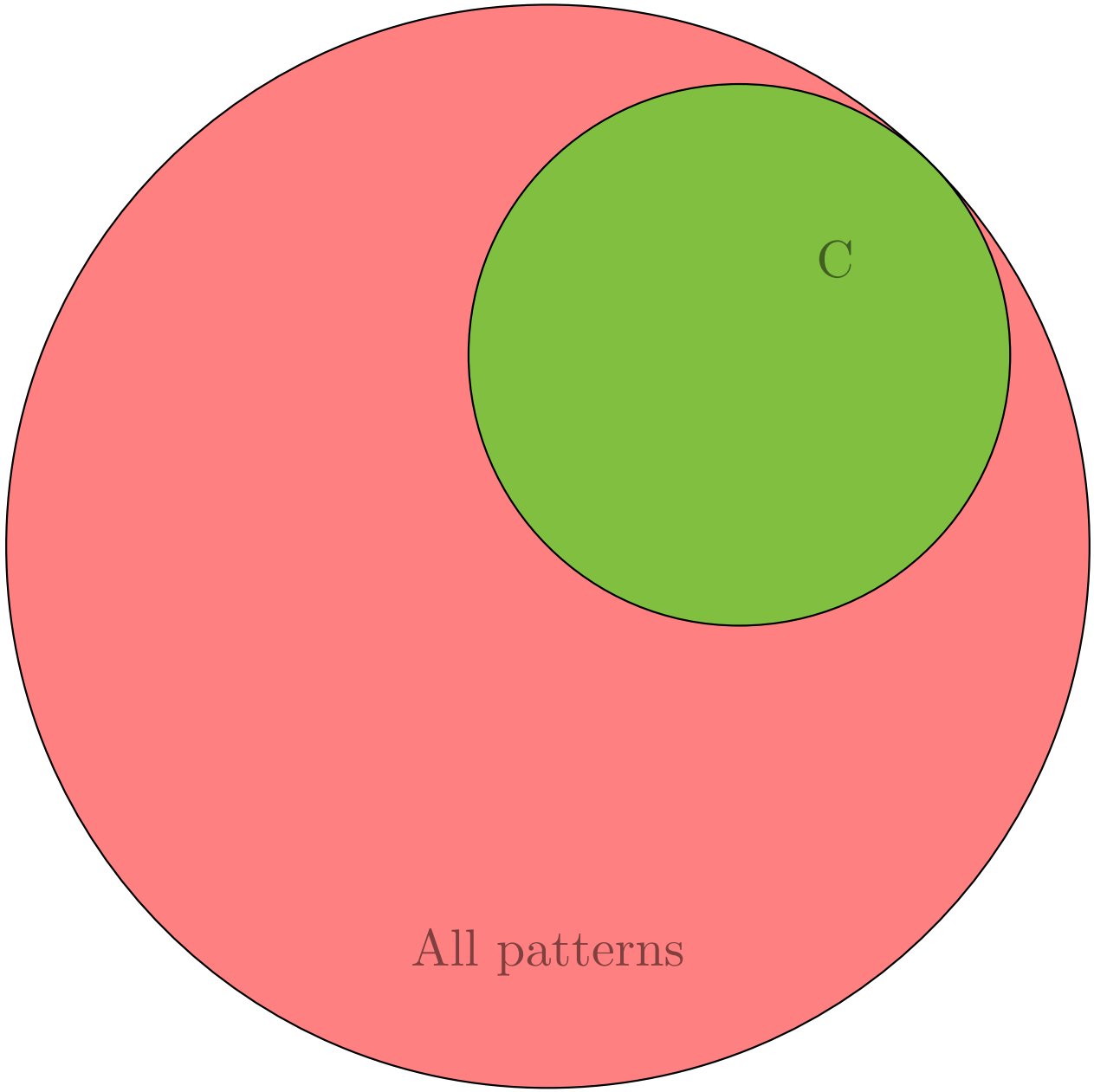


The Perpetual Misconception Machine

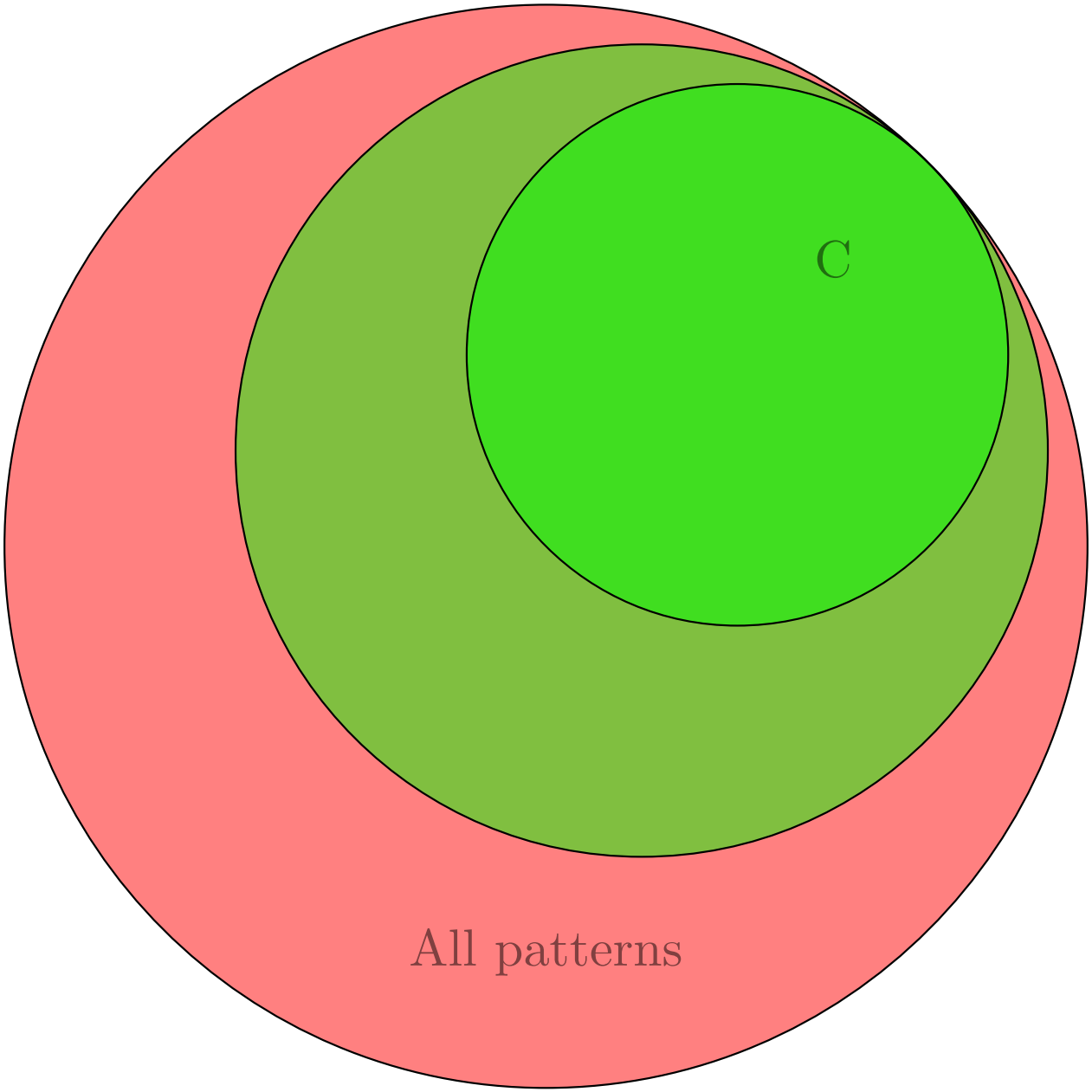
When you believe in things that you don't understand
then you suffer.

– Stevie Wonder

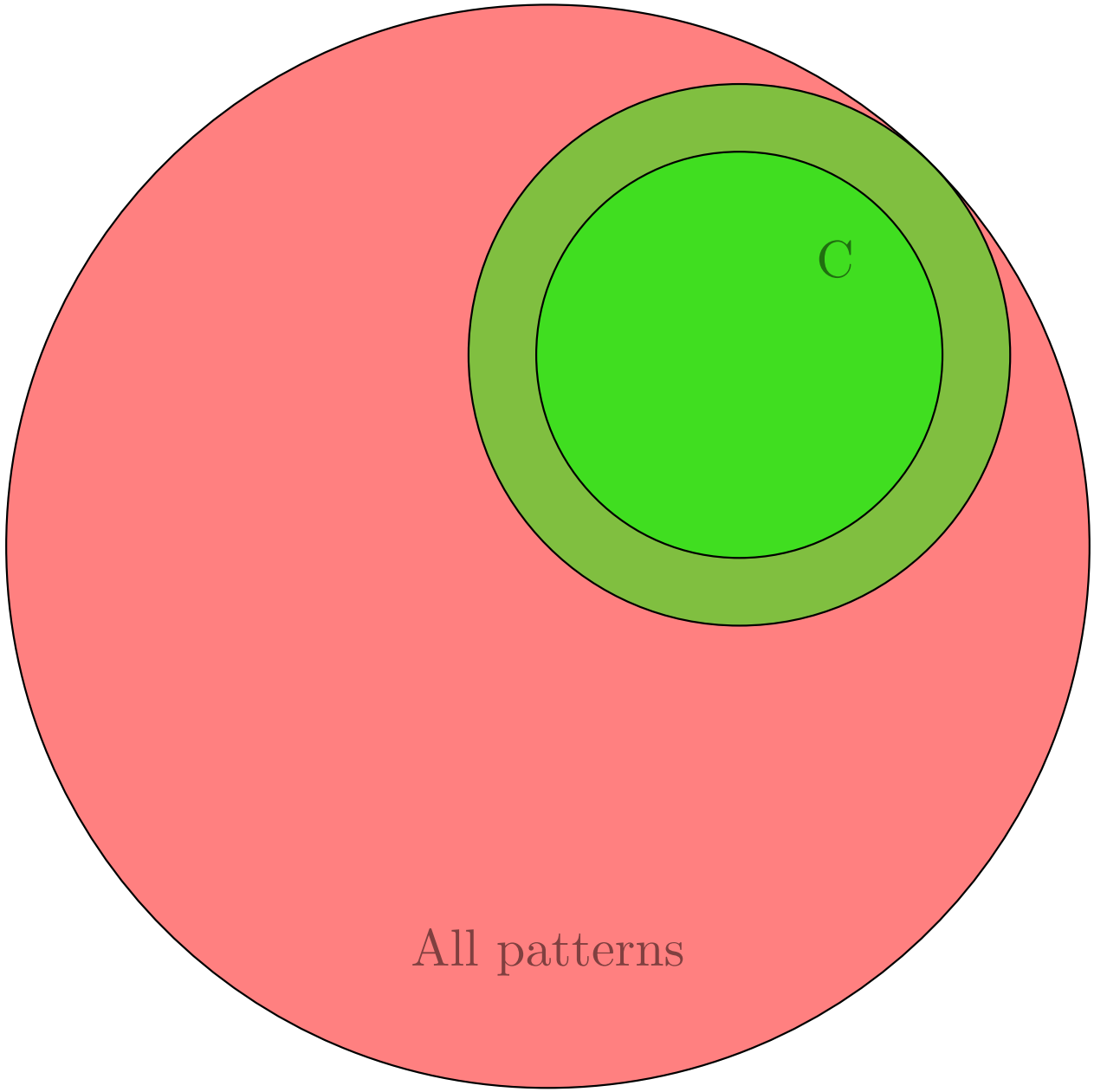
Go smaller, not bigger!



Go smaller, not bigger!



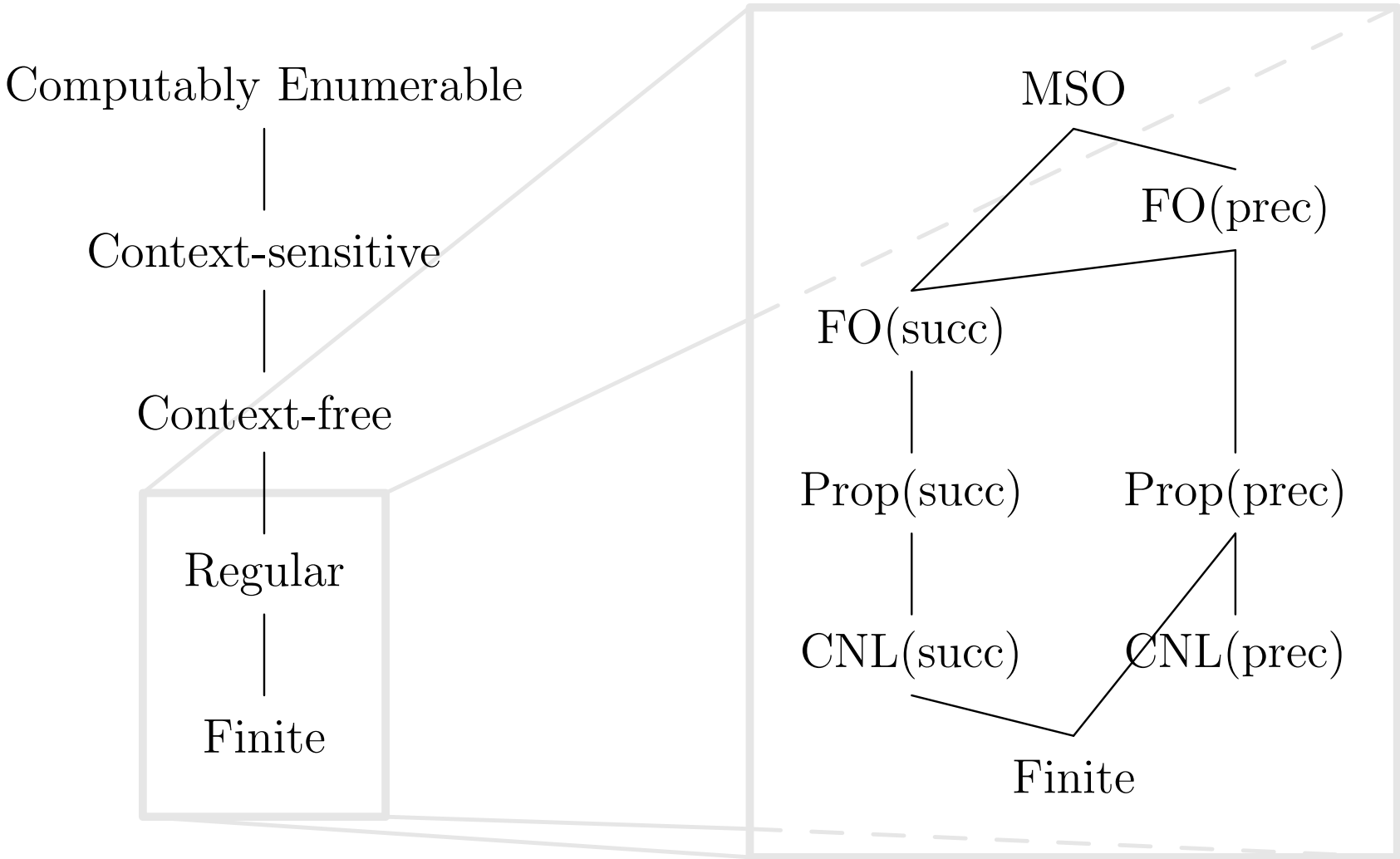
Go smaller, not bigger!



Part III

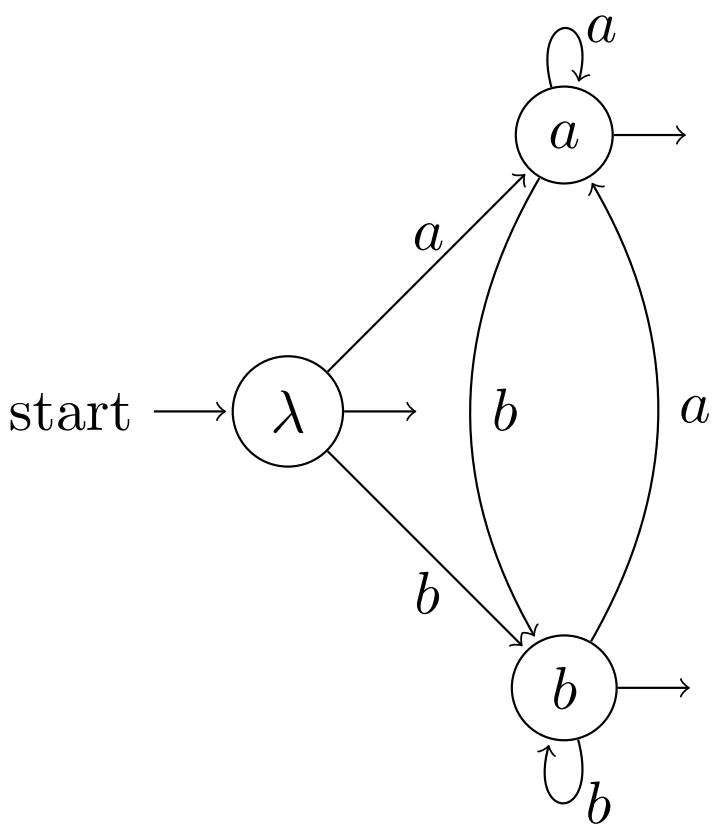
What does ‘going smaller’ look like?

Regular and Subregular Languages



Classes defined by Single DFAs

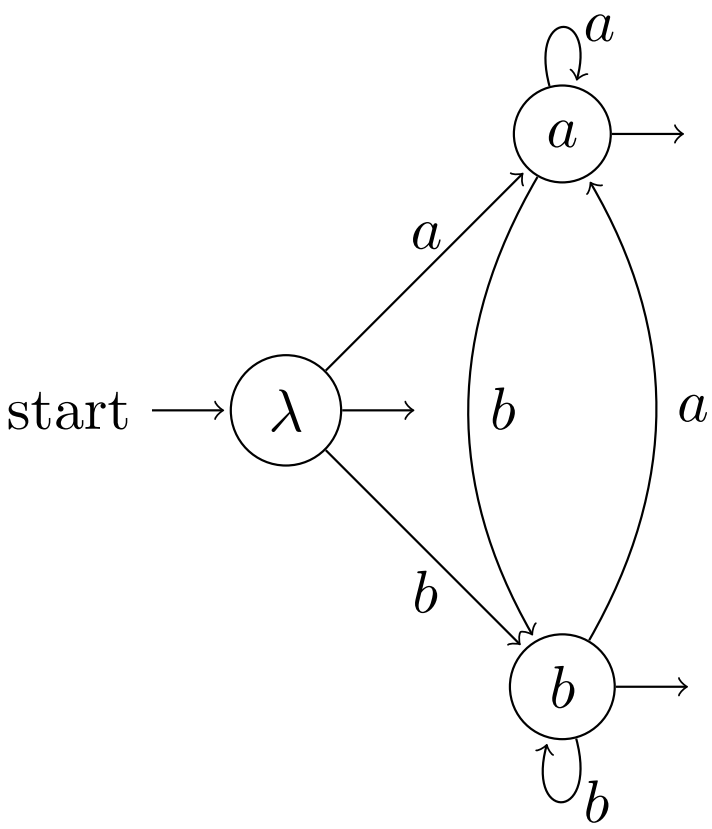
Example: Strictly 2-Local Languages



Parameters
$\theta_{\rtimes a}$
$\theta_{\rtimes b}$
$\theta_{\rtimes \ltimes}$
θ_{aa}
θ_{ab}
$\theta_{a \ltimes}$
θ_{ba}
θ_{bb}
$\theta_{b \ltimes}$

Classes defined by Single DFAs

Example: Strictly 2-Local Languages

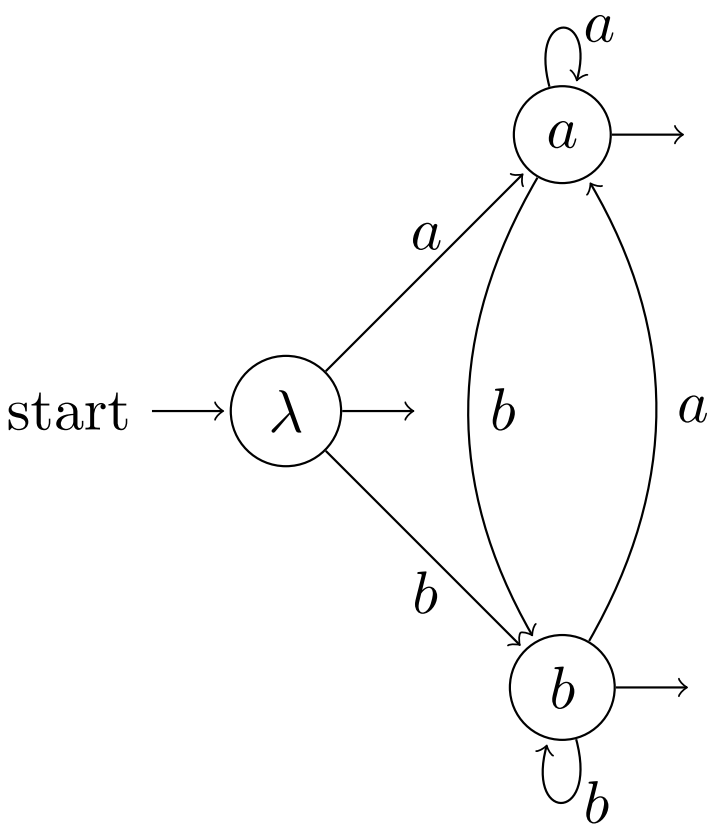


Parameters
$\theta_{\rtimes a}$
$\theta_{\rtimes b}$
$\theta_{\rtimes \ltimes}$
θ_{aa}
θ_{ab}
$\theta_{a \ltimes}$
θ_{ba}
θ_{bb}
$\theta_{b \ltimes}$

$$D = \langle ab, aabb \rangle$$

Classes defined by Single DFAs

Example: Strictly 2-Local Languages



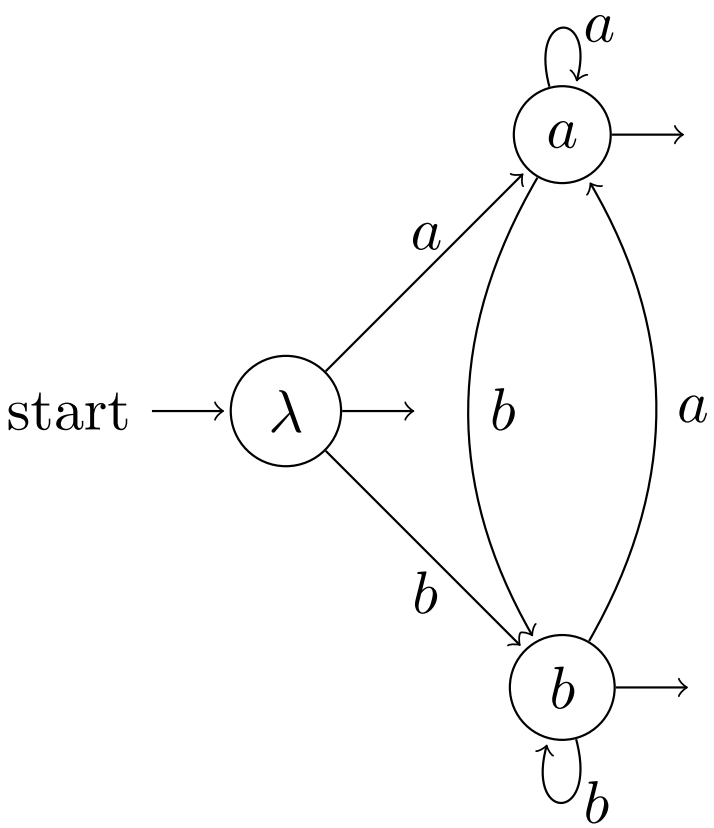
Parameters	
$\theta_{\rtimes a}$	1
$\theta_{\rtimes b}$	
$\theta_{\rtimes \ltimes}$	
θ_{aa}	
θ_{ab}	1
$\theta_{a \ltimes}$	
θ_{ba}	
θ_{bb}	
$\theta_{b \ltimes}$	1

$$D = \langle ab, aabb \rangle$$

Smallest language consistent with D in C is obtained by passing D through DFA and 'activating' parsed transitions. (Heinz and Rogers 2013)

Classes defined by Single DFAs

Example: Strictly 2-Local Languages



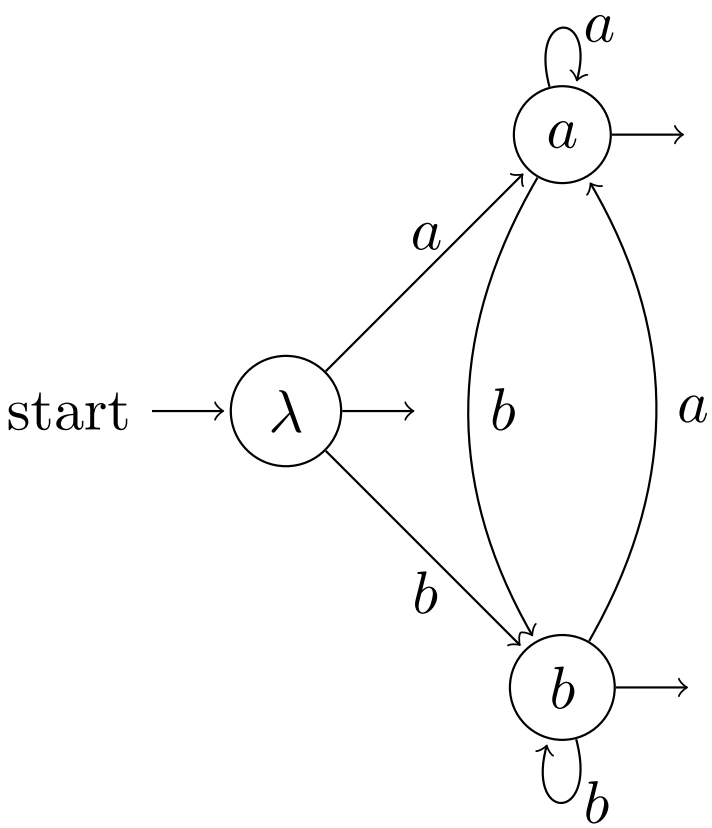
Parameters	
$\theta_{\rtimes a}$	1
$\theta_{\rtimes b}$	
$\theta_{\rtimes \ltimes}$	
θ_{aa}	1
θ_{ab}	1
$\theta_{a \ltimes}$	
θ_{ba}	
θ_{bb}	1
$\theta_{b \ltimes}$	1

$$D = \langle ab, aabb \rangle$$

Smallest language consistent with D in C is obtained by passing D through DFA and 'activating' parsed transitions. (Heinz and Rogers 2013)

Classes defined by Single DFAs

Example: Strictly 2-Local Languages



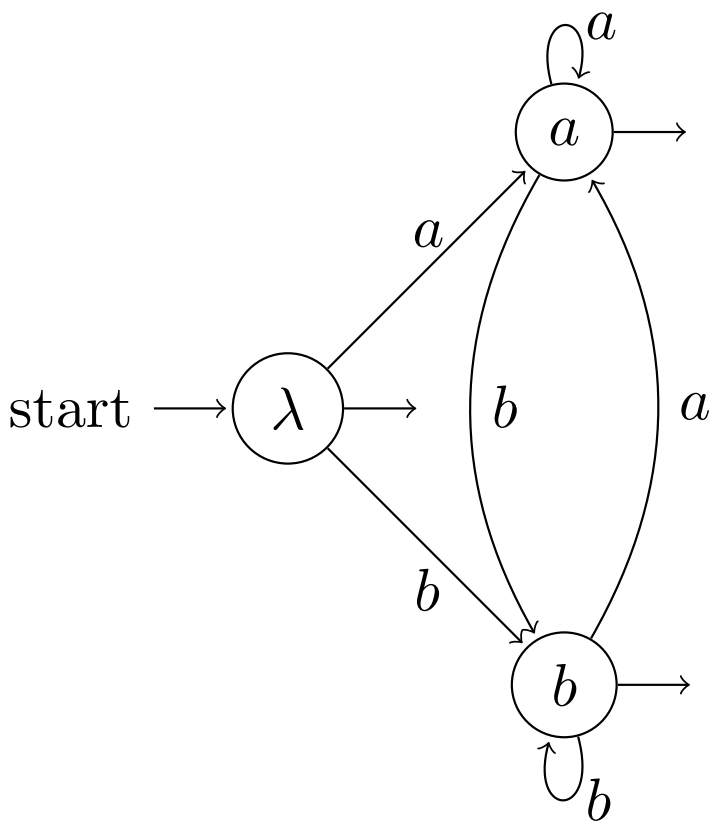
Parameters	
$\theta_{\rtimes a}$	1
$\theta_{\rtimes b}$	0
$\theta_{\rtimes \ltimes}$	0
θ_{aa}	1
θ_{ab}	1
$\theta_{a \ltimes}$	0
θ_{ba}	0
θ_{bb}	1
$\theta_{b \ltimes}$	1

$$D = \langle ab, aabb \rangle$$

Smallest language consistent with D in C is obtained by passing D through DFA and 'activating' parsed transitions. (Heinz and Rogers 2013)

Classes defined by Single DFAs

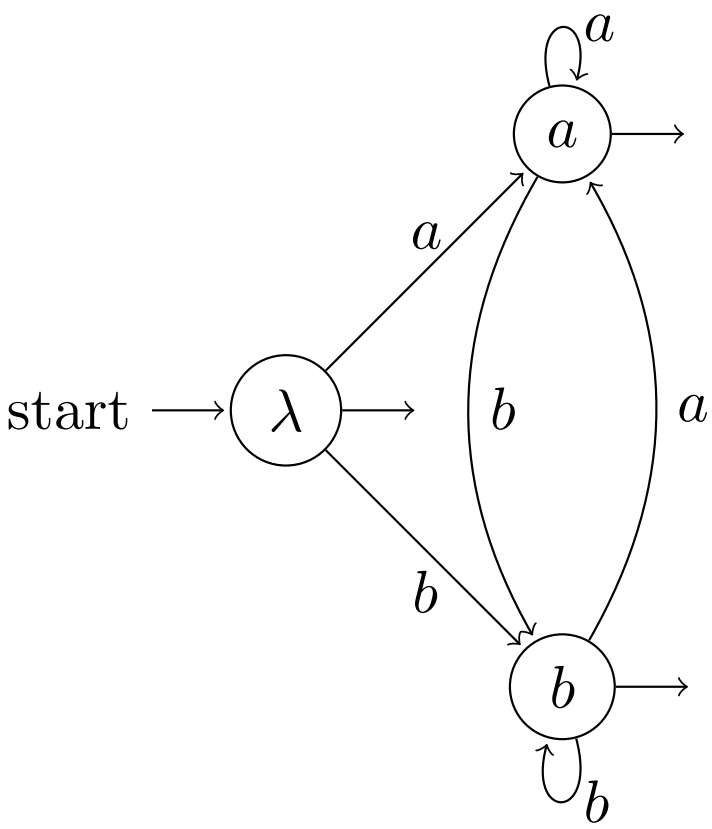
Example: Bigram model



Parameters
$\theta_{\rtimes a}$
$\theta_{\rtimes b}$
$\theta_{\rtimes \ltimes}$
θ_{aa}
θ_{ab}
$\theta_{a \ltimes}$
θ_{ba}
θ_{bb}
$\theta_{b \ltimes}$

Classes defined by Single DFAs

Example: Bigram model

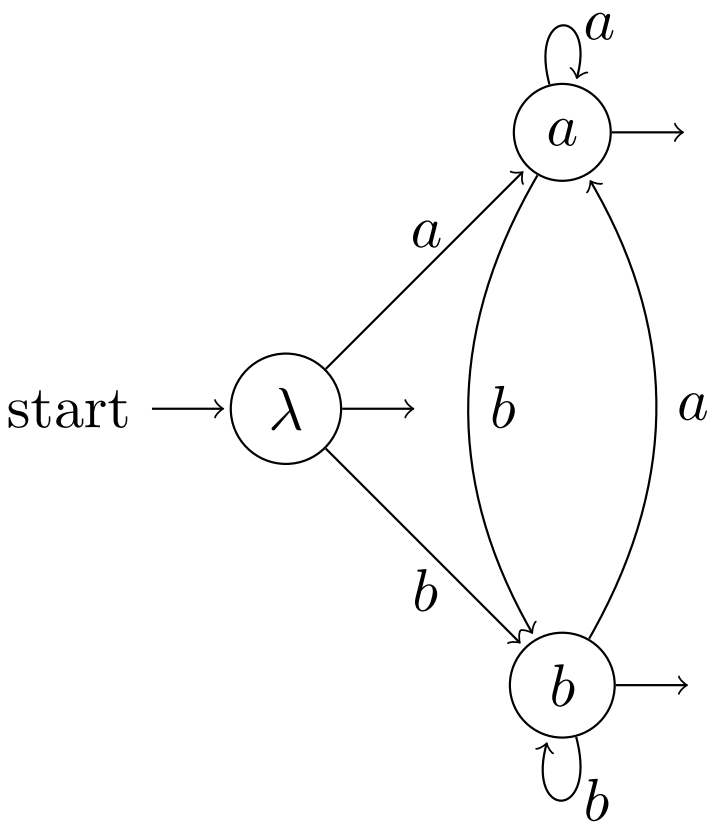


Parameters
$\theta_{\rtimes a}$
$\theta_{\rtimes b}$
$\theta_{\rtimes \ltimes}$
θ_{aa}
θ_{ab}
$\theta_{a \ltimes}$
θ_{ba}
θ_{bb}
$\theta_{b \ltimes}$

$$D = \langle ab, aabb \rangle$$

Classes defined by Single DFAs

Example: Bigram model



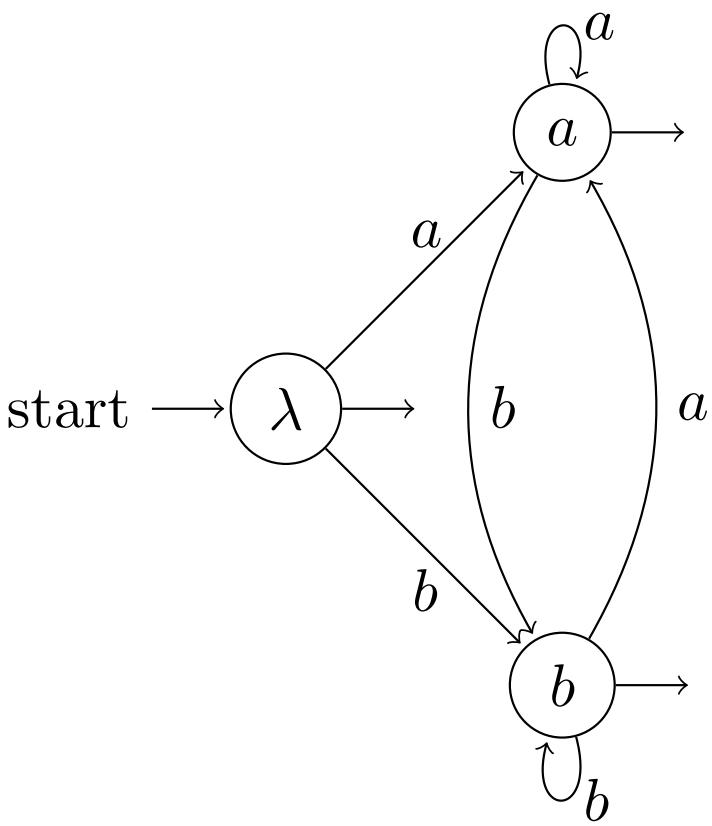
Parameters	
$\theta_{\rtimes a}$	1
$\theta_{\rtimes b}$	
$\theta_{\rtimes \ltimes}$	
θ_{aa}	
θ_{ab}	1
$\theta_{a \ltimes}$	
θ_{ba}	
θ_{bb}	
$\theta_{b \ltimes}$	1

$$D = \langle ab, aabb \rangle$$

Maximum Likelihood Estimate (MLE) is obtained by passing D through DFA and normalizing.
(Vidal et al. 2005)

Classes defined by Single DFAs

Example: Bigram model



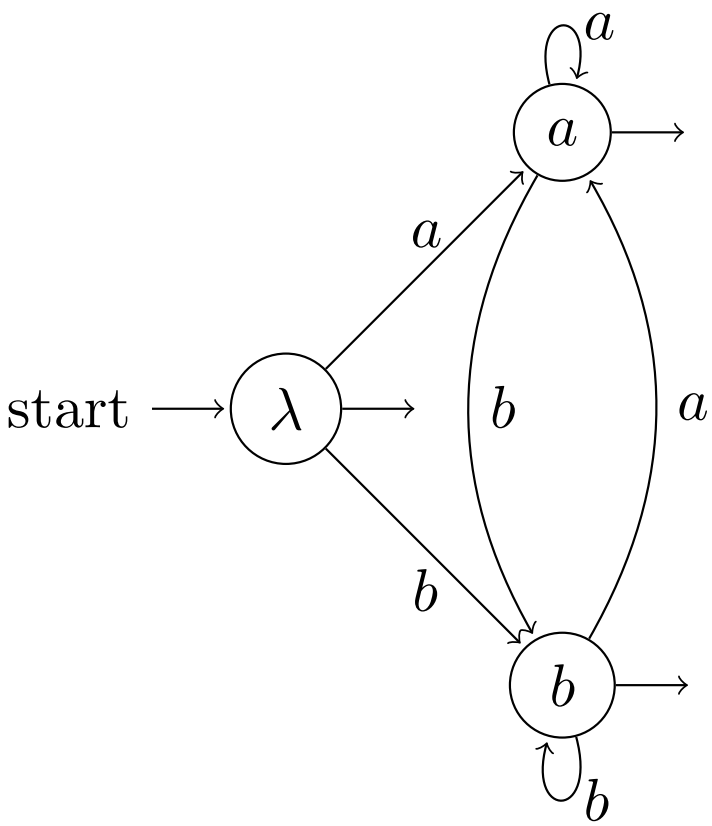
Parameters	
$\theta_{\rtimes a}$	2
$\theta_{\rtimes b}$	
$\theta_{\rtimes \ltimes}$	
θ_{aa}	1
θ_{ab}	2
$\theta_{a\ltimes}$	
θ_{ba}	
θ_{bb}	1
$\theta_{b\ltimes}$	2

$$D = \langle ab, aabb \rangle$$

Maximum Likelihood Estimate (MLE) is obtained by passing D through DFA and normalizing.
(Vidal et al. 2005)

Classes defined by Single DFAs

Example: Bigram model



Parameters	
$\theta_{\rtimes a}$	1
$\theta_{\rtimes b}$	0
$\theta_{\rtimes \ltimes}$	0
θ_{aa}	1/3
θ_{ab}	2/3
$\theta_{a \ltimes}$	0
θ_{ba}	0
θ_{bb}	1/3
$\theta_{b \ltimes}$	2/3

$$D = \langle ab, aabb \rangle$$

Maximum Likelihood Estimate (MLE) is obtained by passing D through DFA and normalizing.
(Vidal et al. 2005)

Learning Results

		Class C defined with	
		single DFA	**finitely many DFA**
type of language	$f : \Sigma^* \rightarrow \{0, 1\}$		
	$f : \Sigma^* \rightarrow [0, 1]$		

1. For Boolean languages, the learning algorithms return the smallest language in C which includes D.
2. For Stochastic languages, the MLE returns the language in C which maximizes likelihood of D.

(Vidal et al. 2005, Heinz and Rogers 2013, Shibata and Heinz 2019)

Learning Results

		Class C defined with	
		single DFA	**finitely many DFA**
type of language	$f : \Sigma^* \rightarrow \{0, 1\}$	✓	
	$f : \Sigma^* \rightarrow [0, 1]$		

1. For Boolean languages, the learning algorithms return the smallest language in C which includes D.
2. For Stochastic languages, the MLE returns the language in C which maximizes likelihood of D.

(Vidal et al. 2005, Heinz and Rogers 2013, Shibata and Heinz 2019)

Learning Results

		Class C defined with	
		single DFA	**finitely many DFA**
type of language	$f : \Sigma^* \rightarrow \{0, 1\}$	✓	
	$f : \Sigma^* \rightarrow [0, 1]$	✓	

1. For Boolean languages, the learning algorithms return the smallest language in C which includes D.
2. For Stochastic languages, the MLE returns the language in C which maximizes likelihood of D.

(Vidal et al. 2005, Heinz and Rogers 2013, Shibata and Heinz 2019)

Learning Results

		Class C defined with: single DFA	Class C defined with: finitely many DFA
type of language	$f : \Sigma^* \rightarrow \{0, 1\}$	✓	✓
	$f : \Sigma^* \rightarrow [0, 1]$	✓	

1. For Boolean languages, the learning algorithms return the smallest language in C which includes D.
2. For Stochastic languages, the MLE returns the language in C which maximizes likelihood of D.

(Vidal et al. 2005, Heinz and Rogers 2013, Shibata and Heinz 2019)

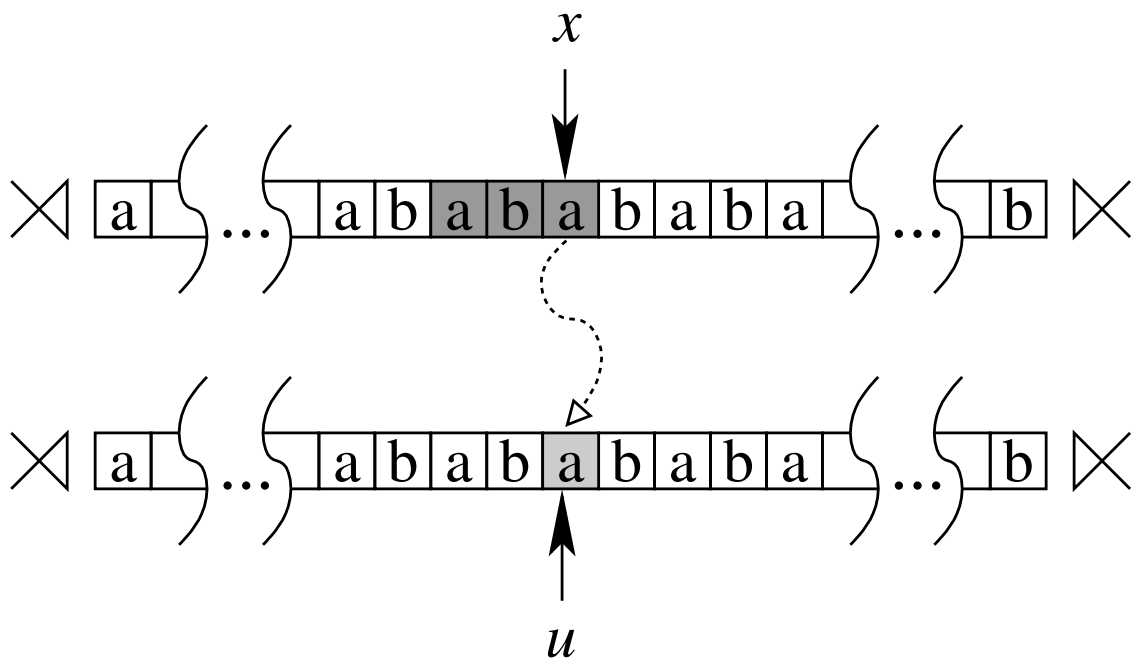
Learning Results

		Class C defined with	
		single DFA	**finitely many DFA**
type of language	$f : \Sigma^* \to \{0, 1\}$	✓	✓
	$f : \Sigma^* \to [0, 1]$	✓	✓

1. For Boolean languages, the learning algorithms return the smallest language in C which includes D.
2. For Stochastic languages, the MLE returns the language in C which maximizes likelihood of D.

(Vidal et al. 2005, Heinz and Rogers 2013, Shibata and Heinz 2019)

Input Strictly Local Functions

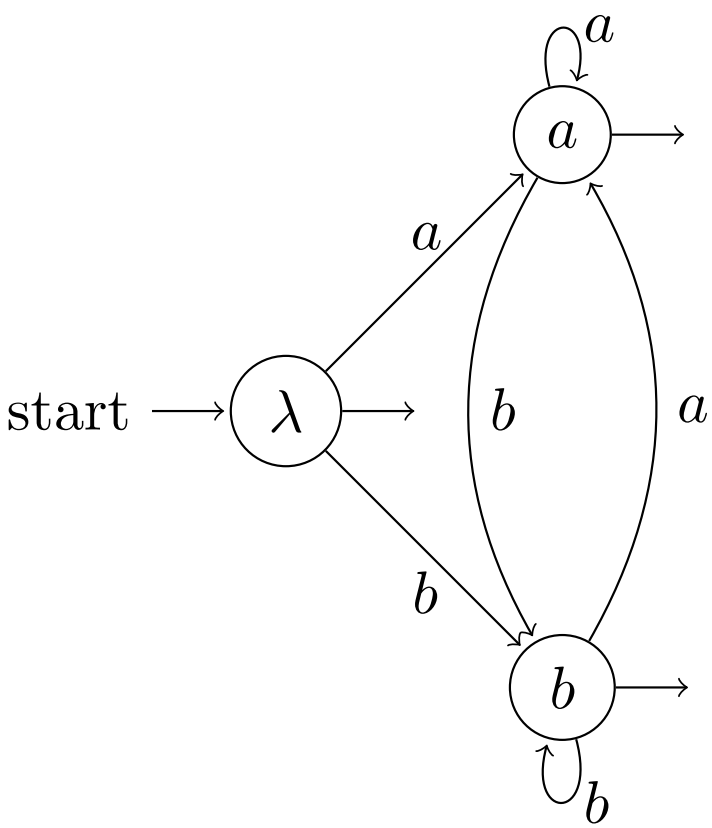


For every Input Strictly 3-Local function, the output string u of each input element x depends only on x and the two input elements previous to x. In other words, the contents of the lightly shaded cell only depends on the contents of the darkly shaded cells.

(Chandlee et al. 2014, 2015, Chandlee and Heinz 2018, Chandlee et al. 2018)

Classes defined by Single DFAs

Example: 2-ISL Model (Parameter values are strings!)



Parameters	
$\theta_{\rtimes a}$	a
$\theta_{\rtimes b}$	b
$\theta_{\rtimes \ltimes}$	$\ltimes$
θ_{aa}	ba
θ_{ab}	b
$\theta_{a\ltimes}$	$\ltimes$
θ_{ba}	a
θ_{bb}	ab
$\theta_{b\ltimes}$	$\ltimes$

$$aabb \mapsto ababab$$

ISL function Learning Results

- Particular finite-state transducers can be used to represent ISL functions.
- Automata-inference techniques (de la Higuera 2010) are used to learn these transducers.

Theorems

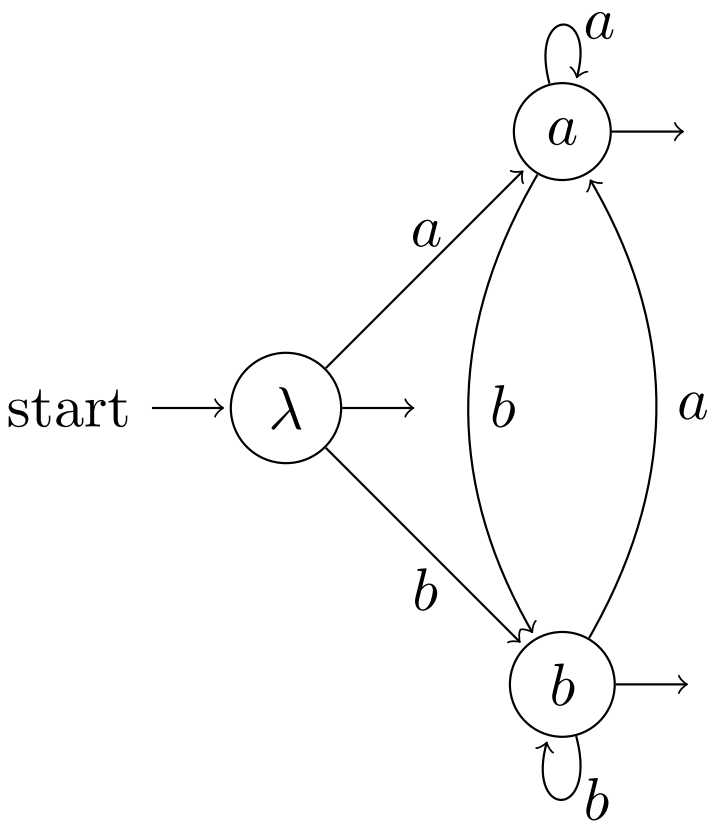
Given k and a sufficient sample of (u, s) pairs any k-ISL function can be *exactly* learned in *polynomial* time and data.

- ISLFLA (quadratic time and data)
- SOSFIA (linear time and data)
- OSLFLA (quadratic time and data)

(Chandlee et al. 2014, 2015, Jardine et al. 2014)

Classes defined by Single DFAs

Example: 2-ISL Model (Parameters are stringsets!)

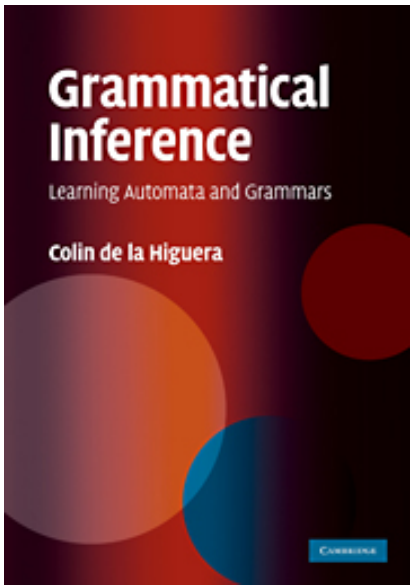


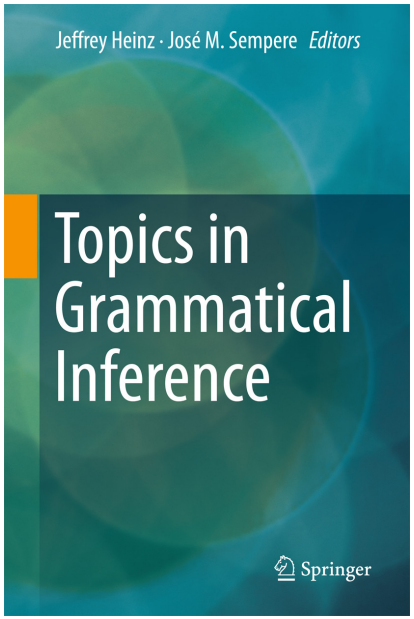
Parameters	
$\theta_{\rtimes a}$	{a}
$\theta_{\rtimes b}$	{b}
$\theta_{\rtimes \ltimes}$	{⋉}
θ_{aa}	{a, ba}
θ_{ab}	{b}
$\theta_{a \ltimes}$	{⋉}
θ_{ba}	{a}
θ_{bb}	{b, ab}
$\theta_{b \ltimes}$	{⋉}

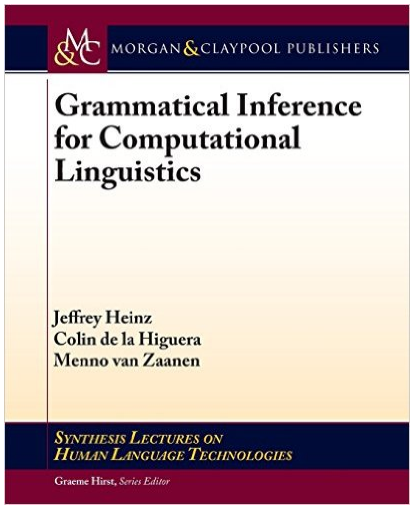
$$aabb \mapsto \{aabb, ababb, aabab, ababab\}$$

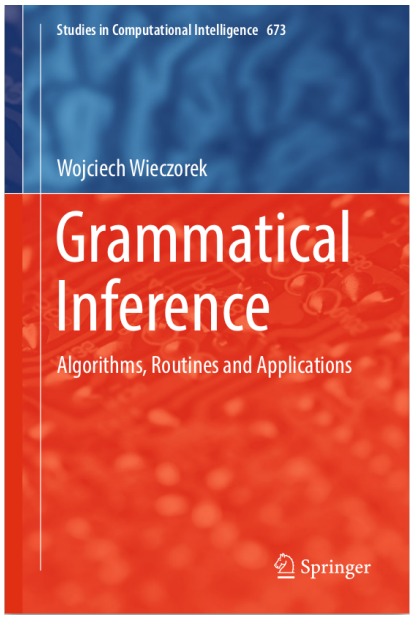
(work in progress with Kiran Eiden and Eric Schieferstein)

Grammatical Inference





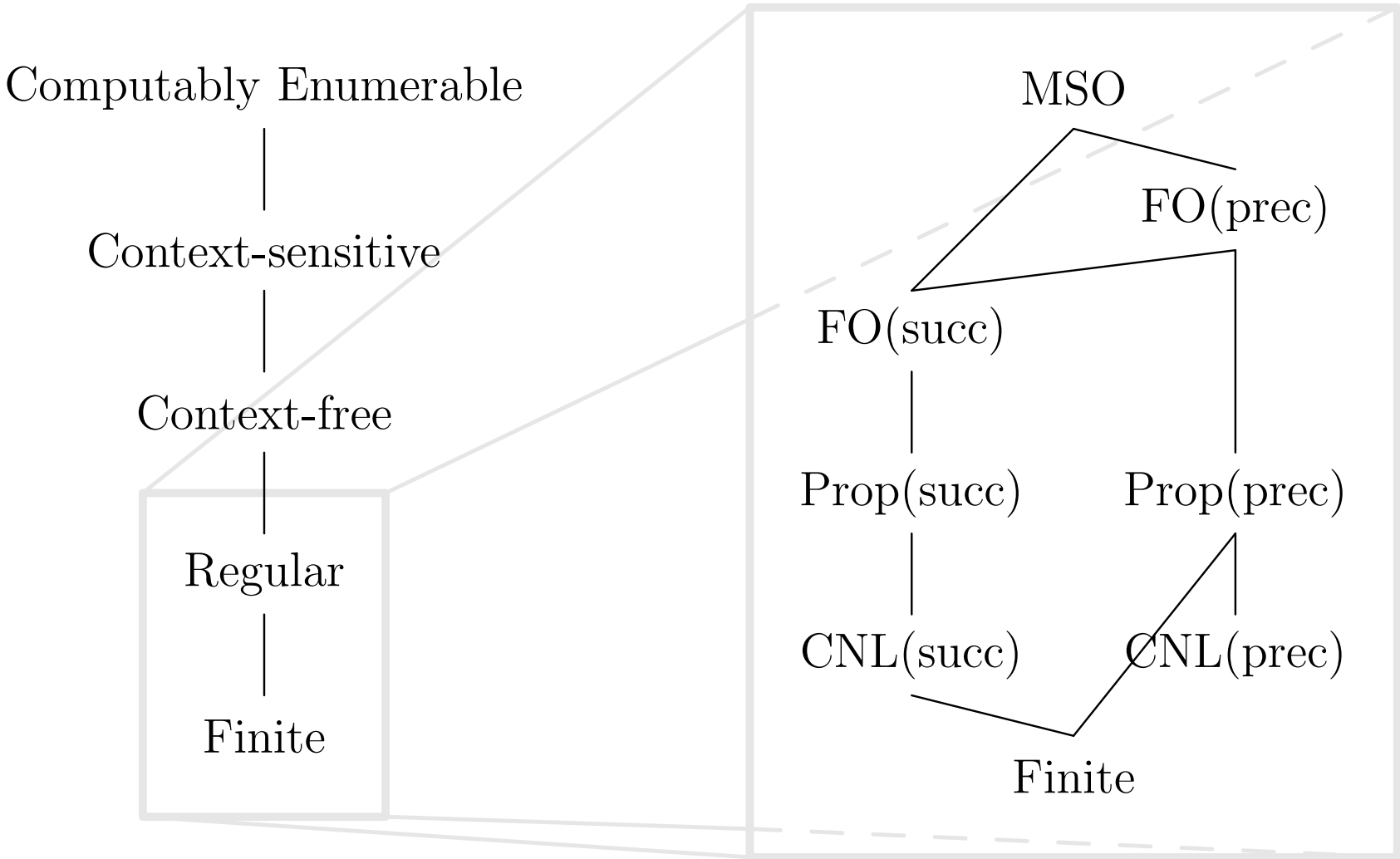




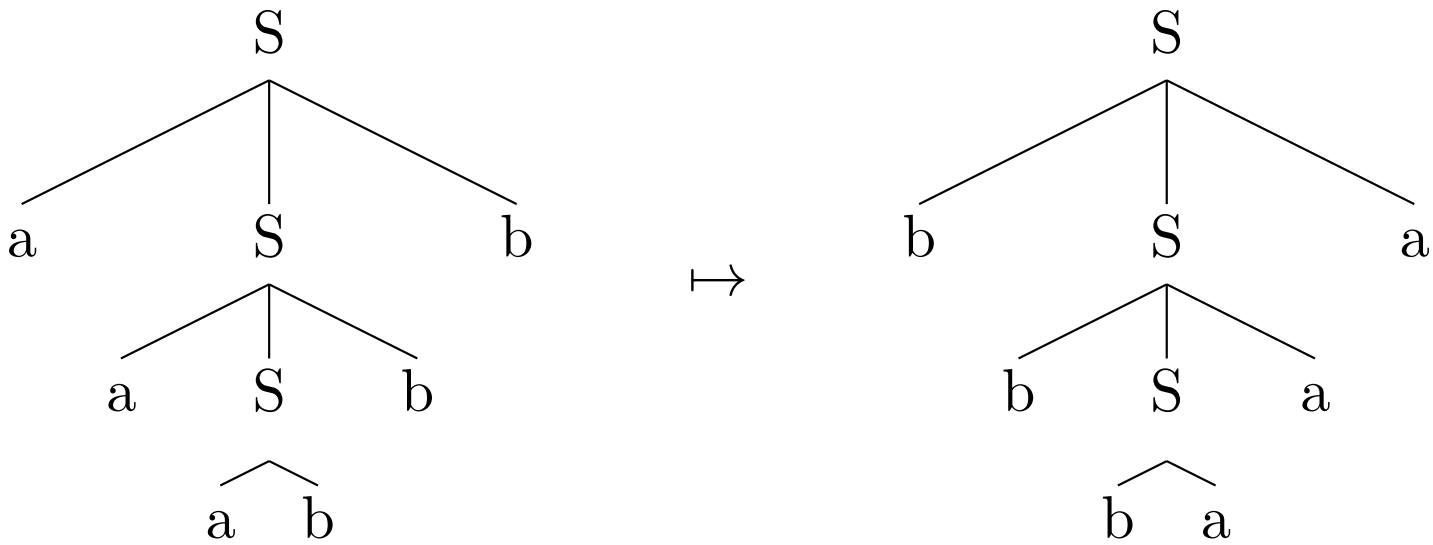
ICGI 2020 in NYC August 26-28!!

`https://grammarlearning.org/`

Regular and Subregular Languages



ISL Bottom-Up Tree Transducers



(Graf 2020, Ji and Heinz 2020)

Part IV

Summing Up

Personal View

- It is a fascinating question how computers–human or machine–can learn anything, and
- How we come by the knowledge of our own language is especially interesting.

Two key ideas

1. Mathematics and theoretical computer science 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!

Appendix

Extra Slides

Learning Results for Regular Languages

1. The class of all DFAs is not identifiable in the limit from positive data (Gold 1967).

Learning Results for Regular Languages

1. The class of all DFAs is not identifiable in the limit from positive data (Gold 1967).
2. It is NP-hard to find the minimal DFA consistent with a finite sample of positive and negative examples (Gold 1978).

Learning Results for Regular Languages

1. The class of all DFAs is not identifiable in the limit from positive data (Gold 1967).
2. It is NP-hard to find the minimal DFA consistent with a finite sample of positive and negative examples (Gold 1978).
3. Each DFA admits a characteristic sample D of positive and negative examples such that RPNI identifies the DFA from any superset of D in cubic time (Oncina and Garcia 1992, DuPont 1996).

Learning Results for Regular Languages

1. The class of all DFAs is not identifiable in the limit from positive data (Gold 1967).
2. It is NP-hard to find the minimal DFA consistent with a finite sample of positive and negative examples (Gold 1978).
3. Each DFA admits a characteristic sample D of positive and negative examples such that RPNI identifies the DFA from any superset of D in cubic time (Oncina and Garcia 1992, DuPont 1996).
4. ALEGRIA/RLIPS (based on RPNI) (Carrasco and Oncina 1994, 1999) learns the class of PDFAs in polynomial time with probability one (de la Higuera and Thollard 2001).

Learning Results for Regular Languages

1. The class of all DFAs is not identifiable in the limit from positive data (Gold 1967).
2. It is NP-hard to find the minimal DFA consistent with a finite sample of positive and negative examples (Gold 1978).
3. Each DFA admits a characteristic sample D of positive and negative examples such that RPNI identifies the DFA from any superset of D in cubic time (Oncina and Garcia 1992, DuPont 1996).
4. ALEGRIA/RLIPS (based on RPNI) (Carrasco and Oncina 1994, 1999) learns the class of PDFAs in polynomial time with probability one (de la Higuera and Thollard 2001).
5. Clark and Thollard (2004) present an algorithm which learns the class of PDFAs in a modified PAC setting. (See also Parekh and Hanover 2001.)

Learning Results for Regular Languages

1. The class of all DFAs is not identifiable in the limit from positive data (Gold 1967).
2. It is NP-hard to find the minimal DFA consistent with a finite sample of positive and negative examples (Gold 1978).
3. Each DFA admits a characteristic sample D of positive and negative examples such that RPNI identifies the DFA from any superset of D in cubic time (Oncina and Garcia 1992, DuPont 1996).
4. ALEGRIA/RLIPS (based on RPNI) (Carrasco and Oncina 1994, 1999) learns the class of PDFAs in polynomial time with probability one (de la Higuera and Thollard 2001).
5. Clark and Thollard (2004) present an algorithm which learns the class of PDFAs in a modified PAC setting. (See also Parekh and Hanover 2001.)
6. Maximization-Expectation techniques are used to learn the class of PNFAs, but there is no guarantee to find a global optimum (Rabiner 1989).