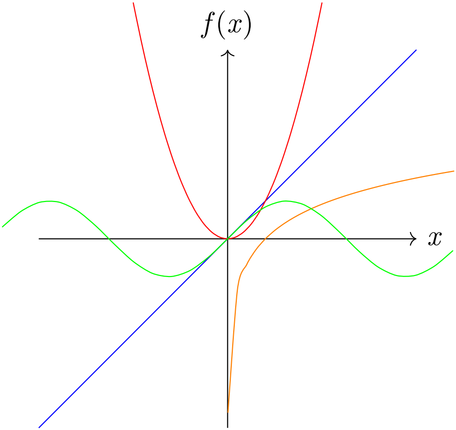
Learning String-to-String Functions

Jeffrey Heinz

Hubert Curien Laboratory
Jean Monnet University
27.10.2023

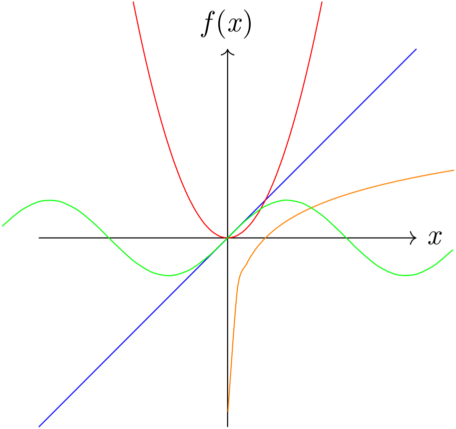


Numbers

- $f(x) = x$
- $f(x) = x^2$
- $f(x) = \ln(x)$
- $f(x) = \sin(x)$

Strings

- $f(x) = x$
- $f(x) = \mathrm{reverse}(x)$
- $f(x) = x \cdot x$
- $f(x) = x^{|x|}$



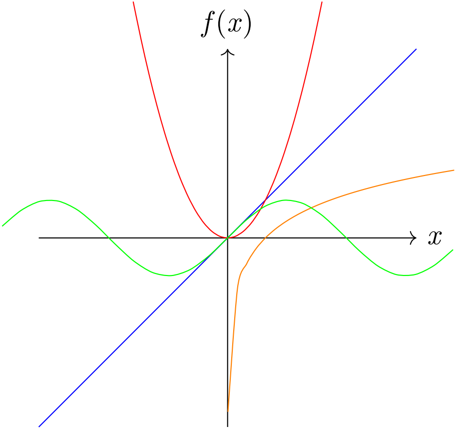
Numbers

- $f(x) = x$
- $f(x) = x^2$
- $f(x) = \ln(x)$
- $f(x) = \sin(x)$

Strings

- $f(x) = x$
  a↦a, ab↦ab, ...
- $f(x) = \text{reverse}(x)$
- $f(x) = x \cdot x$
- $f(x) = x^{|x|}$

Functions



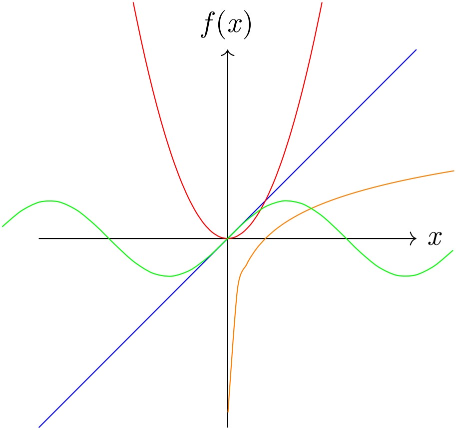
Numbers

- $f(x) = x$
- $f(x) = x^2$
- $f(x) = \ln(x)$
- $f(x) = \sin(x)$

Strings

- $f(x) = x$
- $f(x) = \text{reverse}(x)$
  a↦a, ab↦ba, ...
- $f(x) = x \cdot x$
- $f(x) = x^{|x|}$

Functions



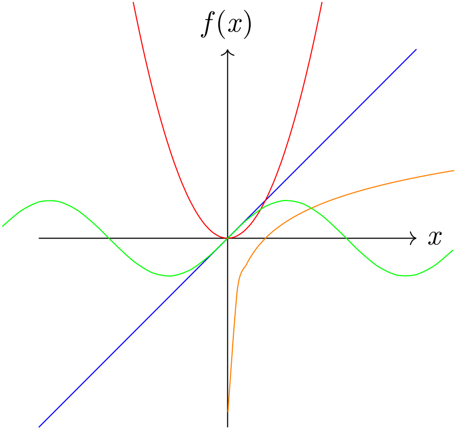
Numbers

- $f(x) = x$
- $f(x) = x^2$
- $f(x) = \ln(x)$
- $f(x) = \sin(x)$

Strings

- $f(x) = x$
- $f(x) = \text{reverse}(x)$
- $f(x) = x \cdot x$
  a$\mapsto$**aa, ab**$\mapsto$**abab, ...**
- $f(x) = x^{|x|}$

Functions



Numbers

- $f(x) = x$
- $f(x) = x^2$
- $f(x) = \ln(x)$
- $f(x) = \sin(x)$

Strings

- $f(x) = x$
- $f(x) = \text{reverse}(x)$
- $f(x) = x \cdot x$
- $f(x) = x^{|x|}$
  a$\mapsto$a, ab$\mapsto$abab, ...

Questions about Functions

1. What kinds of functions are there?
2. What properties do various classes of functions have?
3. How can functions be learned from examples?

Questions about Functions

1. What kinds of functions are there?
2. What properties do various classes of functions have?
3. How can functions be learned from examples?

- Lots is known about numerical functions. Witness the rich vocabulary of classes and relationships. What about string functions?
- Regarding learning, we will see some of the same issues with numerical functions. Given a finite set of points (x, y), there are infinitely many functions that contain them.

Questions about Functions

1. What kinds of functions are there?
2. What properties do various classes of functions have?
3. How can functions be learned from examples?

- Lots is known about numerical functions. Witness the rich vocabulary of classes and relationships. What about string functions?
- Regarding learning, we will see some of the same issues with numerical functions. Given a finite set of points (x, y), there are infinitely many functions that contain them.
- Main message: Attend to smaller, well-structured classes to find feasible learning algorithms. Often, they are enough.

Linguistic Motivation

- Children are exposed to about 10M tokens a year and have a vocabulary of about 1,000 words by age three.
- Yet by this time, their language production largely obeys the grammatical rules of their communities' language.
- Many aspects (not all) of our grammatical knowledge can be expressed as string-to-string functions, especially the **phonological** and **morphological** knowledge.
  - Phonology word final consonant deletion: cf. *vous allez* and *vous voyagez*
  - Morphology inflection: *il prend* and *tu prends*
- How can these kind of patterns be learned from small amounts of data?

Wang 2023 - Experimental Setup

"Learning Transductions and Alignments with RNN Seq2seq Models"

ID	$f(x) = x$	COPY	$f(x) = x \cdot x$
REV	$f(x) = \text{reverse}(x)$	QUADCOPY	$f(x) = x^{\lvert x \rvert}$

- Strings were composed of the 26 lowercase English letters [`a..z`].
- **Train/Dev** data consisted of 1,000 pairs $(x, f(x))$ for each $6 \leq |x| \leq 15$.
- **Test** data consisted of 5,000 pairs $(x, f(x))$ for each $6 \leq |x| \leq 15$.
- **Gen** data consisted of 5,000 pairs $(x, f(x))$ for each $1 \leq |x| \leq 5, 16 \leq |x| \leq 30$.
- Train/Dev/Test/Gen data were pairwise disjoint.

Wang 2023 - Results

Table 1: Aggregate full-sequence accuracy (%) across the four learning tasks for models with various configurations. Best results are in **bold** for the test and gen sets.

		Attentional			Attention-less		
Task	**Dataset**	**SRNN**	**GRU**	**LSTM**	**SRNN**	**GRU**	**LSTM**
Identity	Train	100.00	100.00	100.00	69.74	98.26	100.00
	Test	99.97	**100.00**	**100.00**	42.82	70.46	**77.57**
	Gen	25.52	**37.41**	36.37	0.00	**10.41**	10.01
Rev	Train	100.00	100.00	100.00	100.00	100.00	100.00
	Test	**99.98**	99.87	99.88	**99.55**	88.46	92.85
	Gen	**40.14**	23.54	25.79	**23.89**	19.72	12.42
Total Red	Train	100.00	100.00	99.99	15.22	90.57	93.51
	Test	99.71	**99.77**	99.64	5.60	50.76	**55.17**
	Gen	**42.34**	23.23	20.31	0.00	4.39	**6.18**
Quad Copy	Train	2.43	79.84	82.73	1.62	49.29	67.29
	Test	1.99	67.75	**73.89**	0.61	27.76	**38.03**
	Gen	1.36	**8.20**	6.07	0.00	**0.85**	0.18
Average	Train	75.61	94.96	95.68	46.65	84.53	90.19
	Test	75.41	91.85	**93.35**	37.15	59.36	**65.91**
	Gen	**27.34**	23.10	22.13	5.97	**8.85**	7.20

"We find that RNN seq2seq models are only able to approximate a mapping that fits the training or in-distribution data, instead of learning the underlying functions."

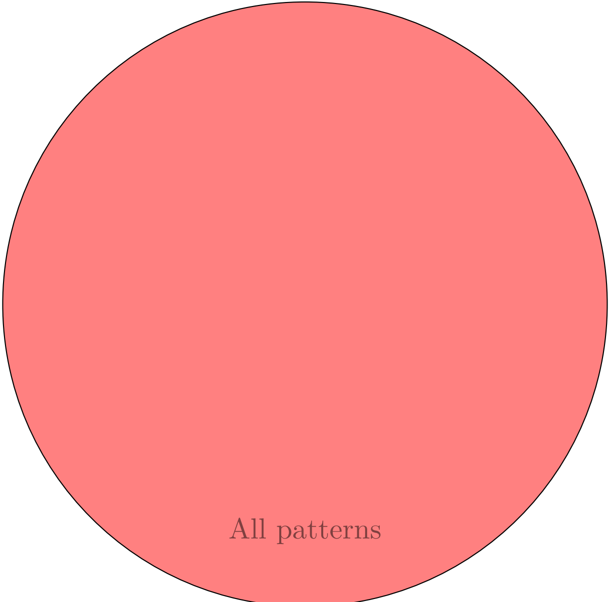
Wang 2023 - Results

Table 1: Aggregate full-sequence accuracy (%) across the four learning tasks for models with various configurations. Best results are in **bold** for the test and gen sets.

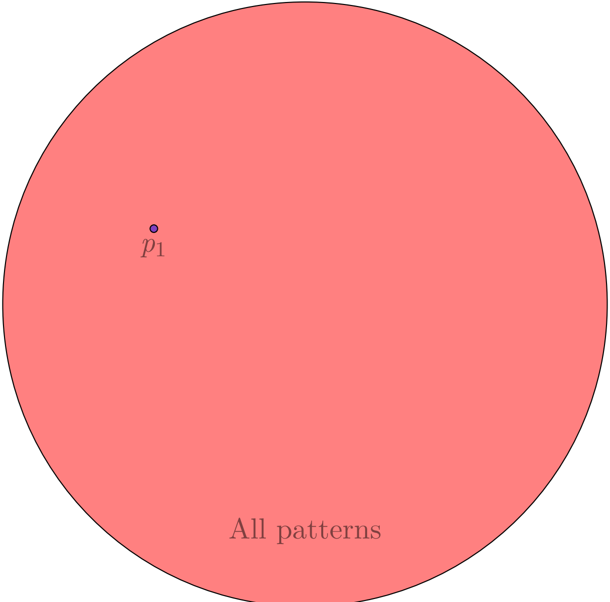
		Attentional			Attention-less		
Task	**Dataset**	**SRNN**	**GRU**	**LSTM**	**SRNN**	**GRU**	**LSTM**
Identity	Train	100.00	100.00	100.00	69.74	98.26	100.00
	Test	99.97	**100.00**	**100.00**	42.82	70.46	**77.57**
	Gen	25.52	**37.41**	36.37	0.00	**10.41**	10.01
Rev	Train	100.00	100.00	100.00	100.00	100.00	100.00
	Test	**99.98**	99.87	99.88	**99.55**	88.46	92.85
	Gen	**40.14**	23.54	25.79	**23.89**	19.72	12.42
Total Red	Train	100.00	100.00	99.99	15.22	90.57	93.51
	Test	99.71	**99.77**	99.64	5.60	50.76	**55.17**
	Gen	**42.34**	23.23	20.31	0.00	4.39	**6.18**
Quad Copy	Train	2.43	79.84	82.73	1.62	49.29	67.29
	Test	1.99	67.75	**73.89**	0.61	27.76	**38.03**
	Gen	1.36	**8.20**	6.07	0.00	**0.85**	0.18
Average	Train	75.61	94.96	95.68	46.65	84.53	90.19
	Test	75.41	91.85	**93.35**	37.15	59.36	**65.91**
	Gen	**27.34**	23.10	22.13	5.97	**8.85**	7.20

"We find that RNN seq2seq models are only able to approximate a mapping that fits the training or in-distribution data, instead of learning the underlying functions."

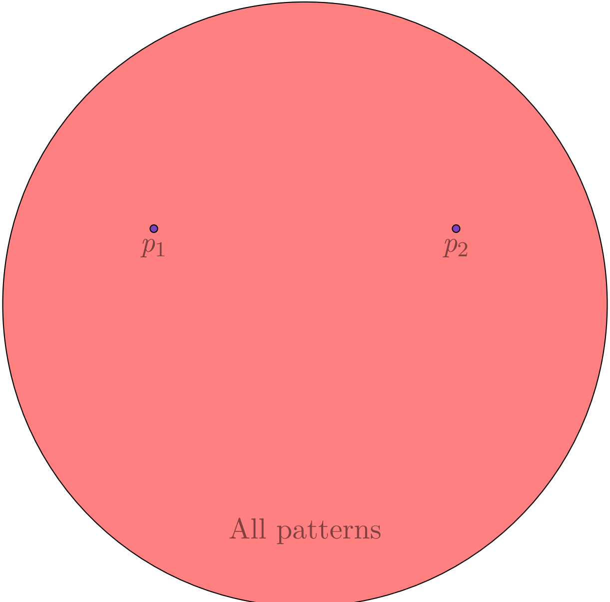
The big picture



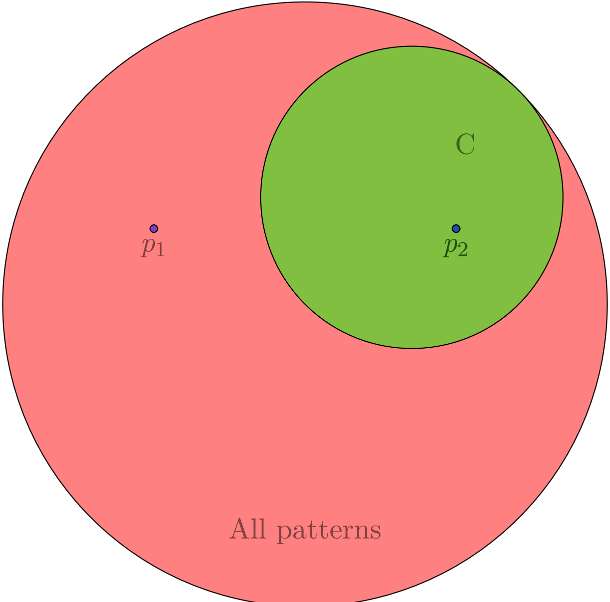
The big picture



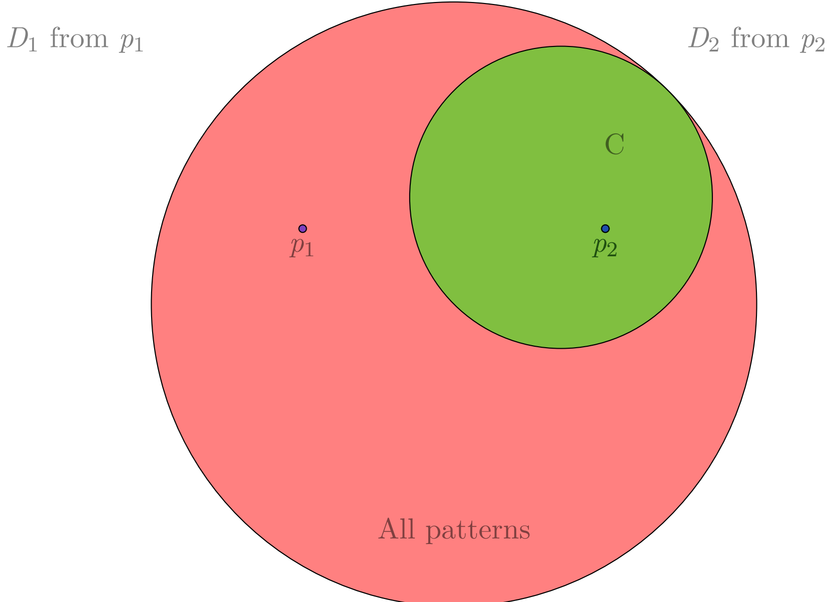
The big picture



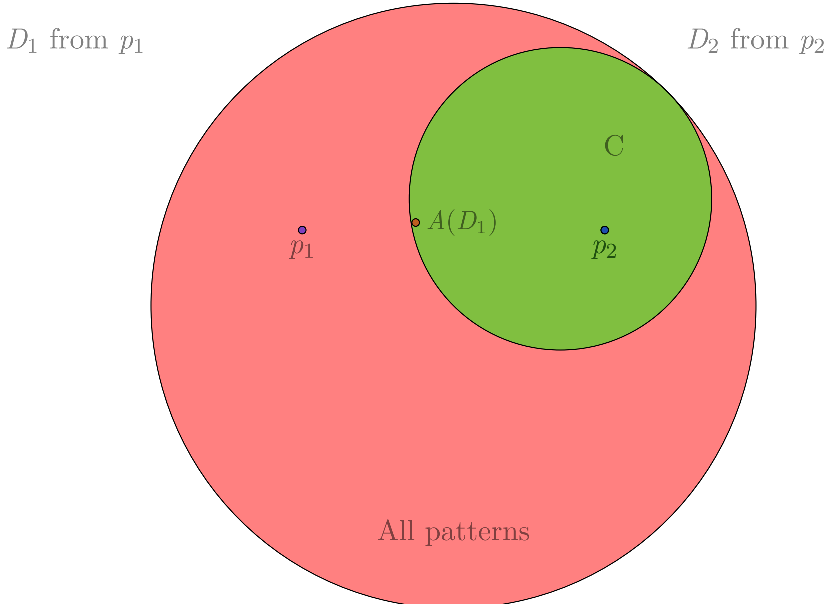
The big picture

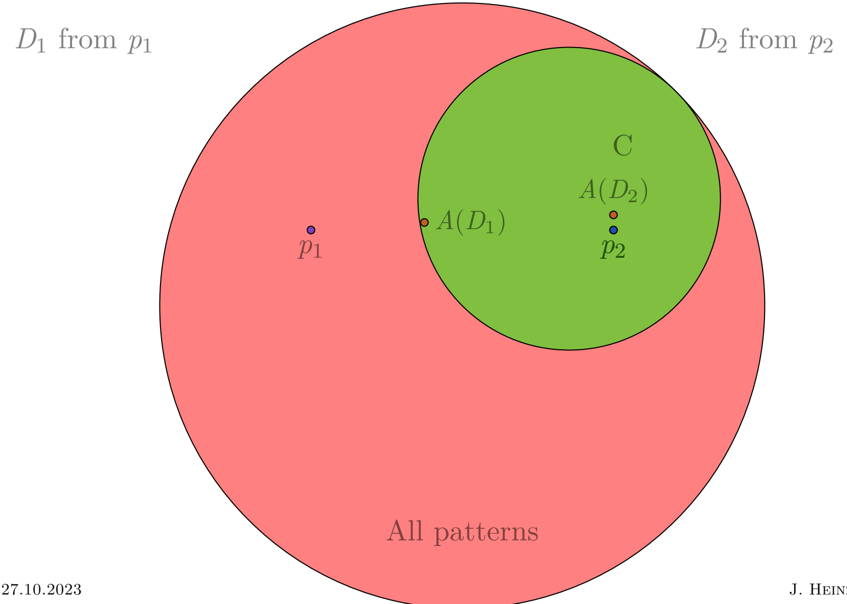


The big picture



The big picture




D_1 from p_1
D_2 from p_2
C
$A(D_2)$
$A(D_1)$
p_1
p_2
All patterns

Part II

Classifying String Functions

Many logically possible string functions

- Reversal
  abcd $\mapsto$ dcba
- Prefixation
  abbb $\mapsto$ cabbb
- Suffixation Mod 2
  abbaab $\mapsto$ abbaabd
  abaab $\mapsto$ abaabc
- Bounded Spreading
  abbb $\mapsto$ aabb
- Unbounded Spreading
  abbb $\mapsto$ aaaa
  abbbdb $\mapsto$ aaaadb
- Projected Unbounded Spreading
  $c^{10}ac^{10}bc^{10}bc^{10}bc^{10}$
  $\mapsto c^{10}ac^{10}ac^{10}ac^{10}ac^{10}$
- Sour Grapes
  abbbb $\mapsto$ aaaaa
  abbdb $\mapsto$ abbdb
- Two-sided Unbounded Spread
  abba $\mapsto$ aaaa
  abbb $\mapsto$ abbb
- Majority Rules
  abbaa $\mapsto$ aaaaa
  abbab $\mapsto$ bbbbb
- Partial Copying
  abcd $\mapsto$ ababcd
- Full Copying
  abcd $\mapsto$ abcdabcd
- Triplication
  abcd $\mapsto$ abcdabcdabcd
- Quadratic Copying
  abcd $\mapsto$ abcdabcdabcdabcd
- Iterated Prefix Copying
  abcd $\mapsto$ a ab abc abcd

Many logically possible string functions

- ✗ Reversal
  abcd ↦ dcba
- ✓ Prefixation
  abbb ↦ cabbb
- ✗ Suffixation Mod 2
  abbaab ↦ abbaabd
  abaab ↦ abaabc
- ✓ Bounded Spreading
  abbb ↦ aabb
- ✓ Unbounded Spreading
  abbb ↦ aaaa
  abbbdb ↦ aaaadb
- ✓ Projected Unbounded Spreading
  $c^{10}ac^{10}bc^{10}bc^{10}bc^{10}$
  $\mapsto c^{10}ac^{10}ac^{10}ac^{10}ac^{10}$
- ✓✗ Sour Grapes
  abbbb ↦ aaaaa
  abbdb ↦ abbdb
- ✓ Two-sided Unbounded Spread
  abba ↦ aaaa
  abbb ↦ abbb
- ✗ Majority Rules
  abbaa ↦ aaaaa
  abbab ↦ bbbbb
- ✓ Partial Copying
  abcd ↦ ababcd
- ✓ Full Copying
  abcd ↦ abcdabcd
- ✓ Triplication
  abcd ↦ abcdabcdabcd
- ✗ Quadratic Copying
  abcd ↦ abcdabcdabcdabcd
- ✗ Iterated Prefix Copying
  abcd ↦ a ab abc abcd

Representing String Functions with Automata

Automata Ingredients

- Read tape for input and write tape for output
- Finite set of states
- Instructions for reading inputs, writing outputs, and changing states

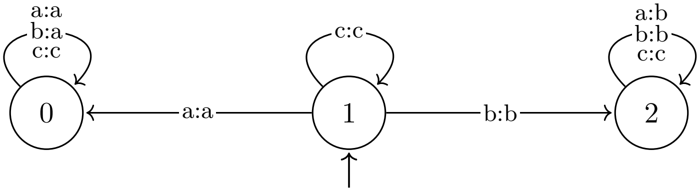
Some Options

- Read 1-way or 2-way
- Deterministic or non-deterministic
- Augment states with stacks, queues, registers

We begin with unaugmented 2-way non-deterministic, and focus on unaugmented 1-way deterministic

Example: long distance harmony

Also known as “projected unbounded spreading”



$$\longrightarrow 1 \xrightarrow[\text{c}]{\text{c}} 1 \xrightarrow[\text{a}]{\text{a}} 0 \xrightarrow[\text{c}]{\text{c}} 0 \xrightarrow[\text{c}]{\text{c}} 0 \xrightarrow[\text{a}]{\text{b}} 0 \cdots \longrightarrow$$

Landscape of Transducers

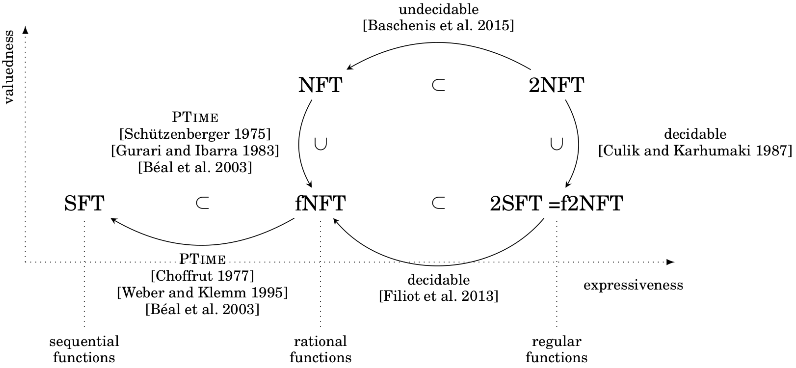


Fig. 3. A landscape of transducers of finite words.

Filiot and Reynier 2016

Landscape of Transducers

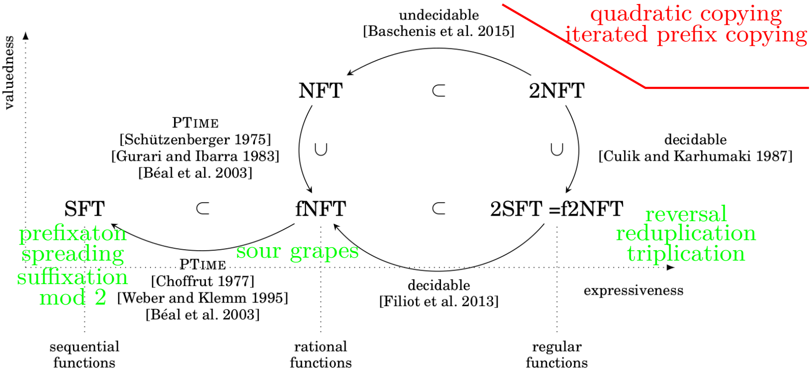


Fig. 3. A landscape of transducers of finite words.

Filiot and Reynier 2016

Algebra provides a finer-grained view

- Inputs to a finite-state automaton affect its behavior in systematic ways.
- The **syntactic monoid** representation helps make its algebraic properties clear.
- There are algorithms to compute it from any 1 way deterministic transducer.

Syntactic Monoids and Semigroups

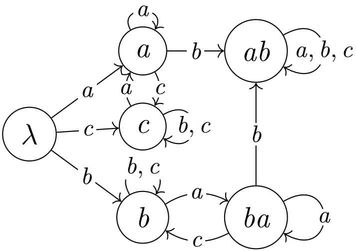
- A semigroup is a set closed under a binary operation $(S, \times)$.

- A monoid is a semigroup with an identity element $(S, \times, 1)$.

- The product of two elements x, y in the syntactic semigroup S of an automaton A is determined by the state reached by taking the path labeled y from state x in A

$$xy = z \text{ iff } x \xhookrightarrow{y} z$$

Example: with $\Sigma = \{a, b, c\}$



	a	b	c	ab	ba
a	a	ab	c	ab	ab
b	ba	b	b	ab	ba
c	a	c	c	ab	a
ab	ab	ab	ab	ab	ab
ba	ba	ab	b	ab	ab

The syntactic monoid of a transducer and its Cayley table.

Definite Structure: $Se = e$

- An idempotent is an element e in a semigroup S such that $ee = e$.
- An automaton is **definite** if and only if its syntactic semigroup has the property that for all idempotents $e \in S$ and for all $x \in S$, it holds that $xe = e$.
- This is often written $Se = e$ with universal quantification left implicit.

	a	b	c	ab	ba
a	a	ab	c	ab	ab
b	ba	b	b	ab	ba
c	a	c	c	ab	a
ab	ab	ab	ab	ab	ab
ba	ba	ab	b	ab	ab

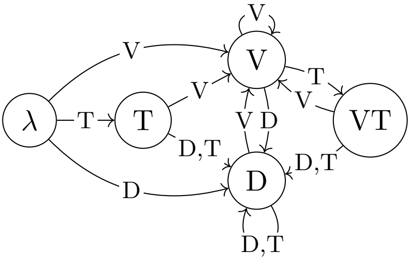
	T	V	D	VT
T	D	V	D	VT
V	VT	V	D	VT
D	D	V	D	VT
VT	D	V	D	VT

Deciding Definiteness

Input: a finite-state automaton

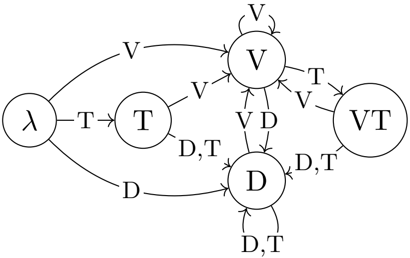
1. Construct the syntactic monoid.
2. Construct the Cayley Table.
3. Identify the idempotents.
4. Return the answer to this question:
   For all idempotents e, does $Se = e$?

Example of Definite Automaton



	T	V	D	VT
T	D	V	D	VT
V	VT	V	D	VT
D	D	V	D	VT
VT	D	V	D	VT

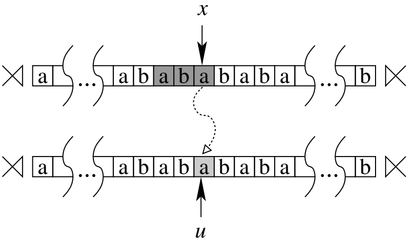
Example of Definite Automaton



	T	V	D	VT
T	D	V	D	VT
V	VT	V	D	VT
D	D	V	D	VT
VT	D	V	D	VT

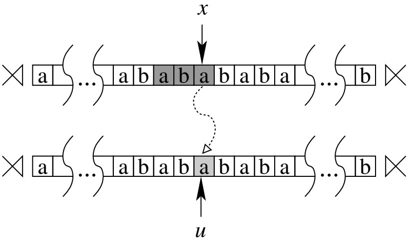
This is the syntactic monoid of a transducer representing **Intervocalic voicing** whereby voiceless consonants become voiced between two vowels, e.g. *tantata* $\mapsto$ *tantada*.

Definite string functions have the property that there exists a k such that what is output at any given point only depends on the last k letters read (cf. the Markov property).



(Perles et al. 1963, Vaysse 1986, Lambert and Heinz 2023)

Definite functions are also known as Input Strictly Local Functions



- Many morphological and phonological processes in natural language are definite functions.
- The k-definite functions are learnable (in a sense to be made more precise in a moment).

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

Other Algebraic Classes

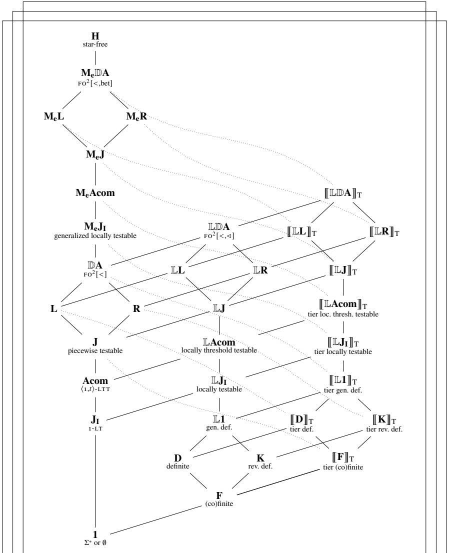
The algebraic classification has been a rich area of study for several decades.

Class	Property
Definite	$Se = e$
Reverse Definite	$eS = e$
Nilpotent	$SeS = e$
Generalized Definite	$eSe = e$
Locally Testable	$\forall x, y \in eSe : xx = x, xy = yx$
...	

Table: Some Algebraic Varieties

Green 1951, Ginzburg 1966, Almeida 1995, Pin 1984, 1997, 2021, a.o.

Lambert 2022, 2023: Algebraic Hierarchy



Some subregular classes for sets and functions over sequences, circa 2022

Part III

Learning Regular Functions

What does learning mean?

Some ideas...

A ML system **is a consistent estimator for a parametric model** iff for each parameter Θ in the model, for every stream of data generated randomly i.i.d. according to Θ, $Pr(\lim_{n \to \infty} \hat{\Theta} = \Theta) = 1$.

What does learning mean?

Some ideas...

A ML system **probably approximately correctly (PAC) learns a class of concepts** iff for each concept C in the class, for each distribution D over the instance space, there is some n such that for all $m > n$, we have $\Pr[\texttt{error}_D(\hat{C}, C) < \epsilon] > 1 - \delta$. (Valiant 1984)

What does learning mean?

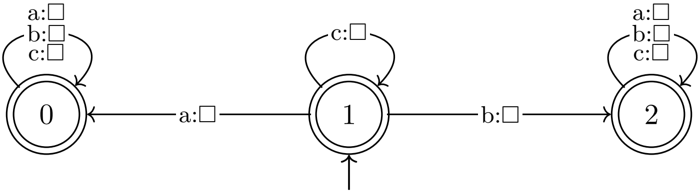
Some ideas...

A ML system **identifies a class of concepts in the limit from positive data** iff for each concept C in the class, for every positive presentation of C, there is some n such that for all $m > n$, we have $R_m = R_n$ and $C(R_n) = C$. (Gold 1967)

Theoretical Learning Results

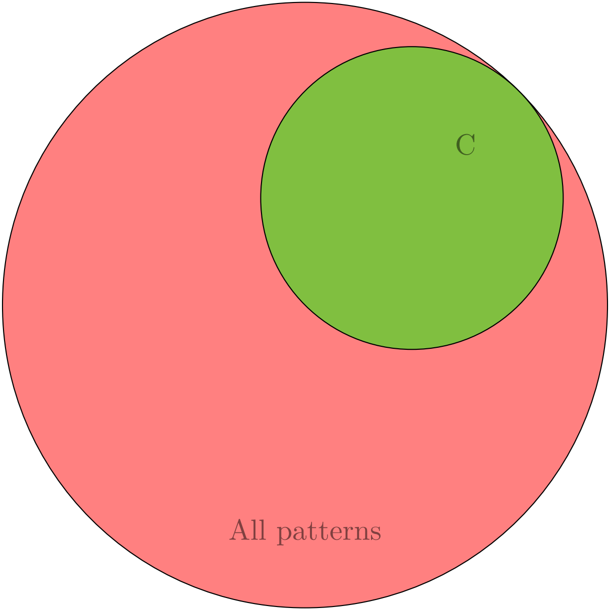
1. The class of rational relations is neither PAC-learnable nor identifiable in the limit from positive data.
2. Sequential functions (1 way deterministic) can be identified in the limit from positive data in cubic time and data by OSTIA (Oncina et al. 1993).
3. Algorithms for identifying definite functions in the limit from positive data run in quadratic time and data (Chandlee et al. 2014) and even in linear time and data (Jardine et al. 2014).

SOSFIA (Jardine et al. 2014)

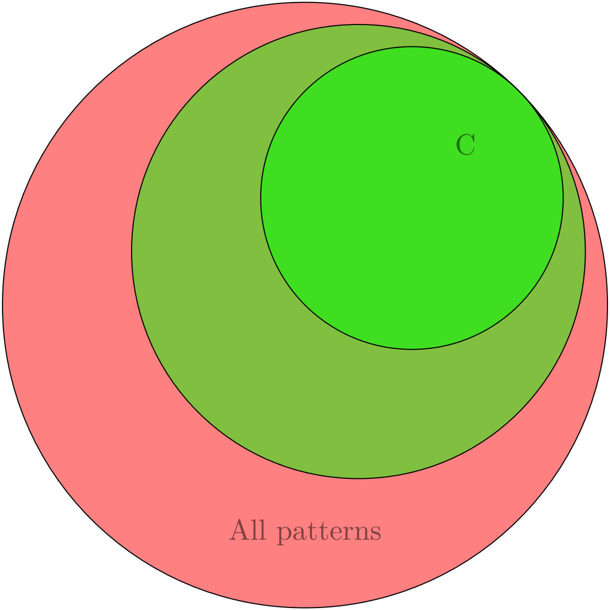


- Many algebraic classes, appropriately parameterized, can be represented by a single deterministic transducer.
- The only difference is how the outputs are labeled.
- Consequently, learning reduces to inferring the output labels of the transitions.
- Jardine et al. 2014 provides a theoretical, not practical, solution to this problem.

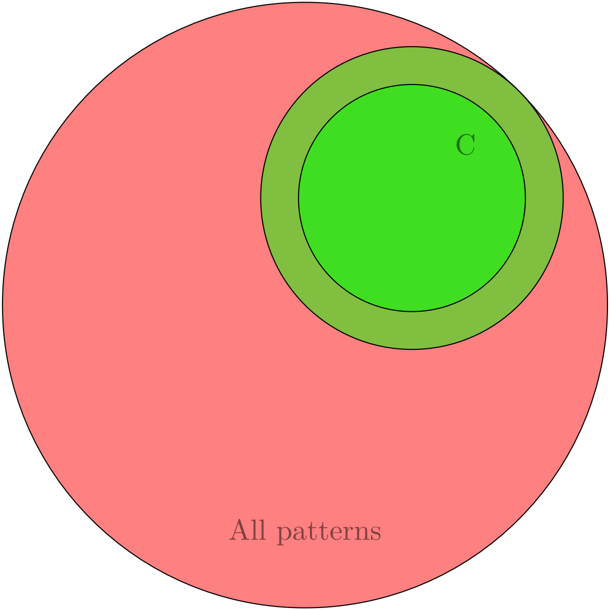
Go smaller, not bigger!



Go smaller, not bigger!



Go smaller, not bigger!



Conclusion

1. The learning of string functions is ongoing.

2. Talking about non-local dependencies is like talking about non-linear functions. There is a rich classification of them, let's use it!

3. Feasibly solving a learning problem requires defining a target class C of patterns which must be suitably structured.

4. It can help reduce the instance space of the learning problem to only consider the kinds of things you have to learn.

Open Questions

1. How can we make SOSFIA more practical?
2. Learning factored representations of transducers
3. Subregular classes of tree transductions for natural language syntax and semantics
4. Deterministic regular relations

Thank You