Grammatical inference and subregular phonology

Adam Jardine
Rutgers University

December 11, 2019 · Tel Aviv University

Review

Outline of course

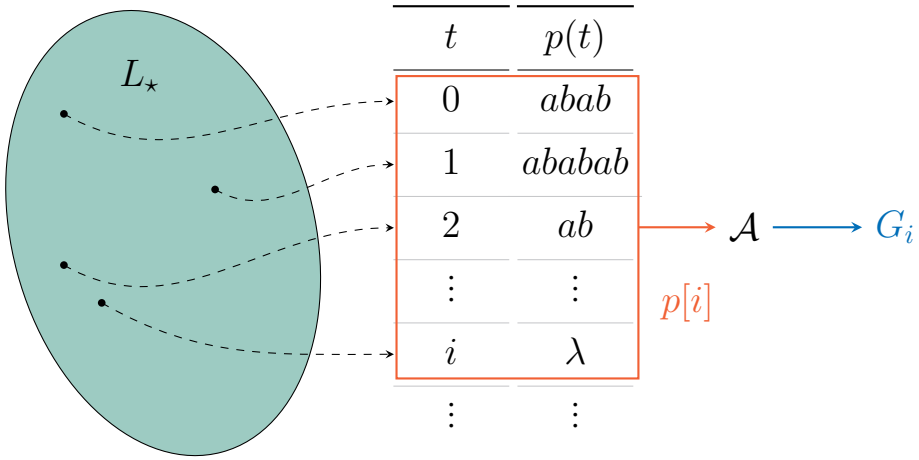
- **Day 1:** Learning, languages, and grammars
- **Day 2:** Learning strictly local grammars
- **Day 3:** Automata and input strictly local functions
- **Day 4:** Learning functions and stochastic patterns, other open questions

Review of days 1 & 2

- Phonological patterns are governed by restrictive computational universals
- We studied one such universal of **strict locality**

Review of days 1 & 2

- We studied learning SL languages under the paradigm of **identification in the limit from positive data**



Today

- Learning with **finite-state automata** for
  - strictly local languages
  - **input-strictly local functions**

Strictly local acceptors

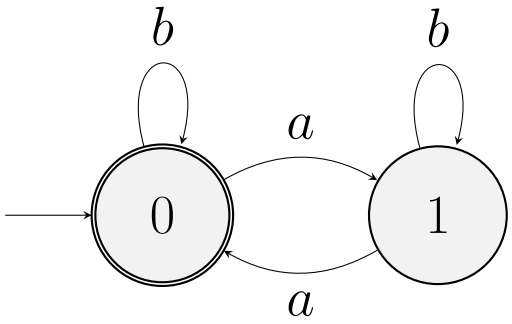
Strictly local acceptors

Engelfriet & Hoogeboom, 2001

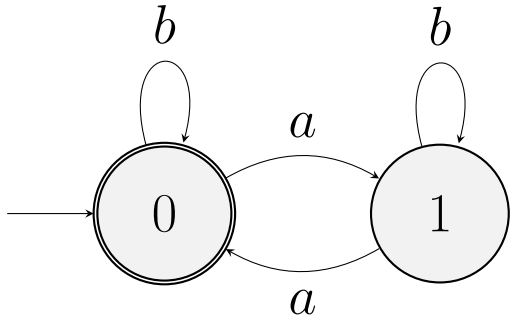
"It is always a pleasant surprise when two formalisms, introduced with different motivations, turn out to be equally powerful, as this indicates that the underlying concept is a natural one." (p. 216)

Strictly local acceptors

- A **finite-state acceptor (FSA)** is a set of **states** and **transitions** between states

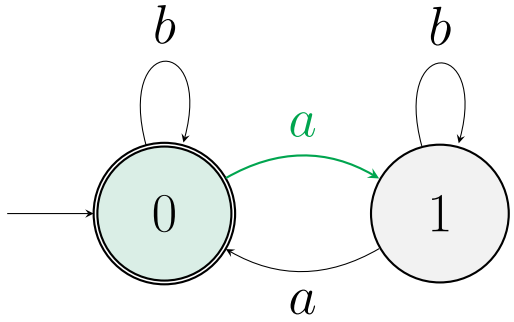


Strictly local acceptors



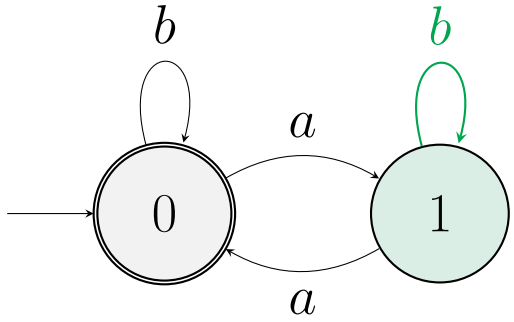
$a\ b\ b\ a$

Strictly local acceptors



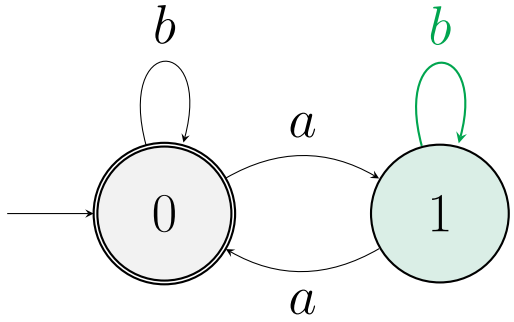
$$\begin{array}{cccc} a & b & b & a \\ 0 & \to & 1 & \end{array}$$

Strictly local acceptors



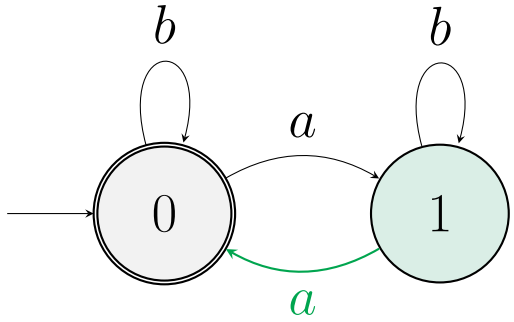
$$\begin{array}{cccccccc} & a & & b & & b & a \\ 0 & \to & 1 & \to & 1 & & \end{array}$$

Strictly local acceptors



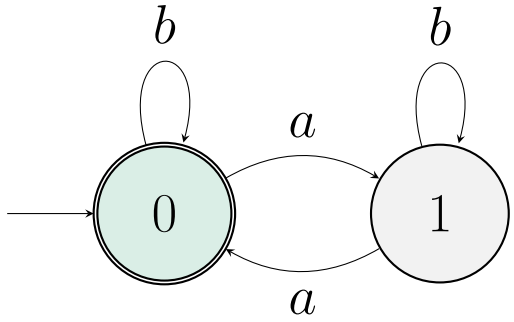
$$\begin{array}{ccccccc} & a & & b & & b & & a \\ 0 & \to & 1 & \to & 1 & \to & 1 \end{array}$$

Strictly local acceptors



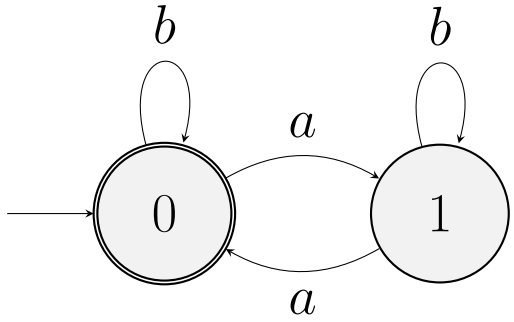
$$\begin{array}{ccccccccc} & & a & & b & & b & & a \\ 0 & \to & 1 & \to & 1 & \to & 1 & \to & 0 \end{array}$$

Strictly local acceptors



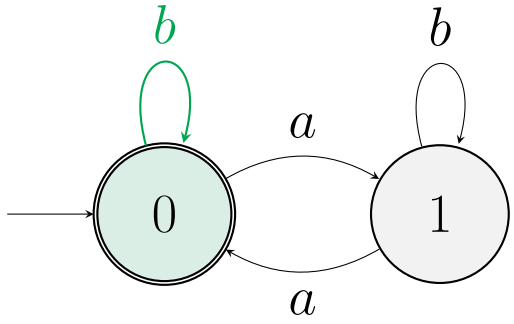
$$\begin{array}{ccccccccccc} & & a & & b & & b & & a & & \\ 0 & \to & 1 & \to & 1 & \to & 1 & \to & 0 & \checkmark \end{array}$$

Strictly local acceptors



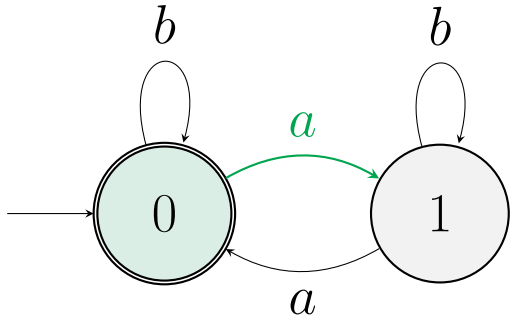
$b\ a\ a\ b\ b\ a$

Strictly local acceptors



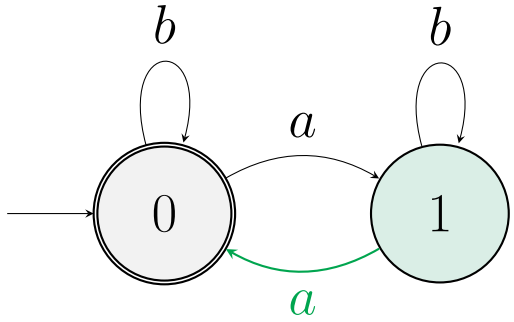
	b	a	a	b	b	a
0	$\rightarrow$	0				

Strictly local acceptors



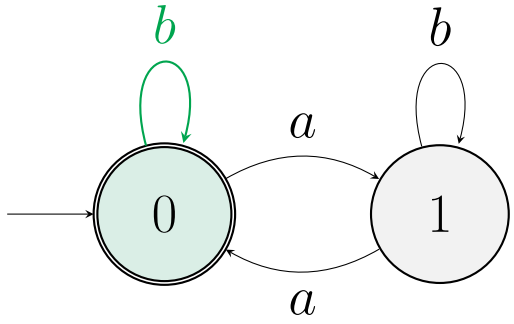
$$\begin{array}{cccccccccc} & b & & a & & a & b & b & a \\ 0 & \to & 0 & \to & 1 & & & & \end{array}$$

Strictly local acceptors



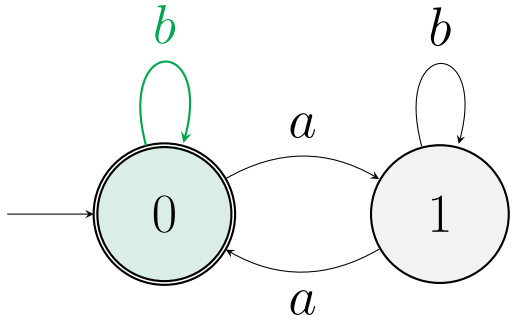
$$\begin{array}{cccccccccc} & b & & a & & a & & b & b & a \\ 0 & \to & 0 & \to & 1 & \to & 0 & & & \end{array}$$

Strictly local acceptors



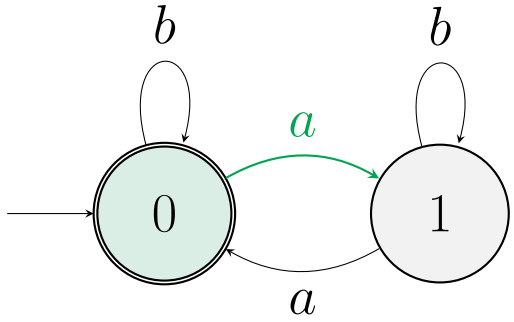
$$\begin{array}{cccccccccc} & b & & a & & a & & b & b & a \\ 0 & \to & 0 & \to & 1 & \to & 0 & \to & 0 & \end{array}$$

Strictly local acceptors



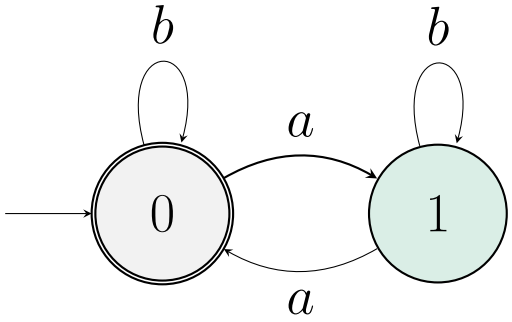
$$\begin{array}{ccccccccccc} & b & & a & & a & & b & & b & & a \\ 0 & \to & 0 & \to & 1 & \to & 0 & \to & 0 & \to & 0 & \end{array}$$

Strictly local acceptors



$$\begin{array}{ccccccccccccc} & & b & & a & & a & & b & & b & & a \\ 0 & \to & 0 & \to & 1 & \to & 0 & \to & 0 & \to & 0 & \to & 1 \end{array}$$

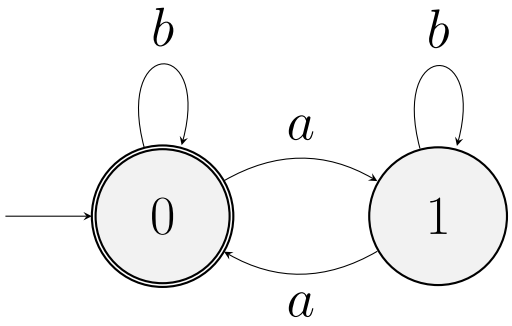
Strictly local acceptors



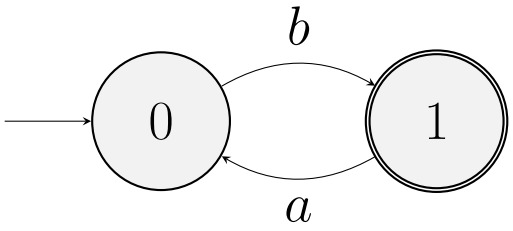
$$\begin{array}{ccccccccccccc} & b & & a & & a & & b & & b & & a & \\ 0 & \to & 0 & \to & 1 & \to & 0 & \to & 0 & \to & 0 & \to & 1 \end{array}$$ ✗

Strictly local acceptors

- A **SL$_k$FSA**'s states represent the $k-1$ factors of Σ^*



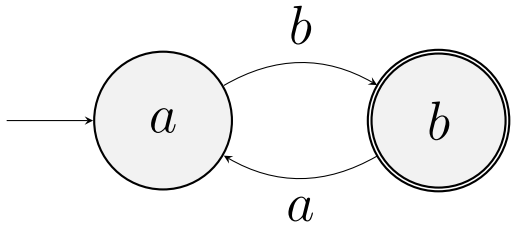
Not SL_k for any k



SL_2; $0 = b$, $1 = a$

Strictly local acceptors

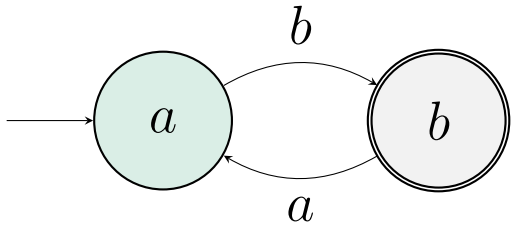
- Traversing a SL_kFSA is equivalent to scanning for k factors

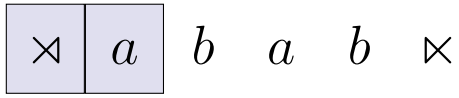


$$\rtimes \quad a \quad b \quad a \quad b \quad \ltimes$$

Strictly local acceptors

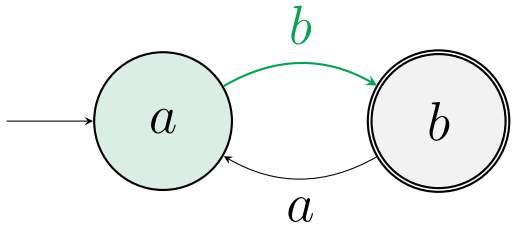
- Traversing a SL_kFSA is equivalent to scanning for k factors

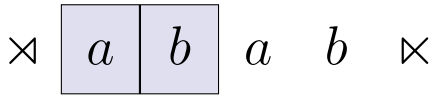




Strictly local acceptors

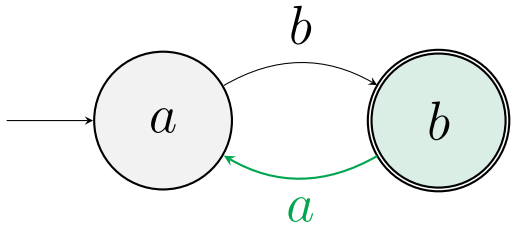
- Traversing a SL_kFSA is equivalent to scanning for k factors

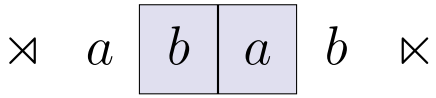




Strictly local acceptors

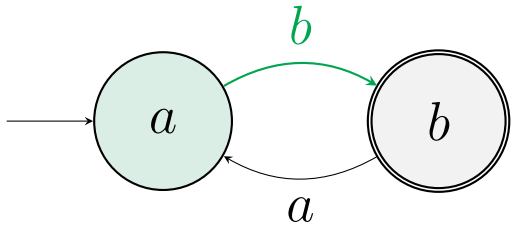
- Traversing a SL_kFSA is equivalent to scanning for k factors

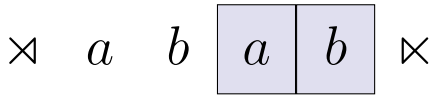




Strictly local acceptors

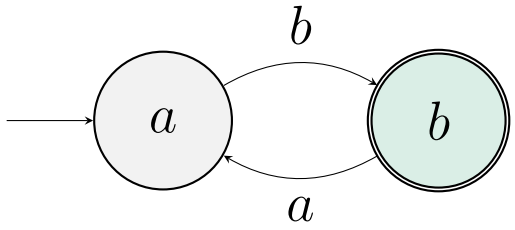
- Traversing a SL_kFSA is equivalent to scanning for k factors

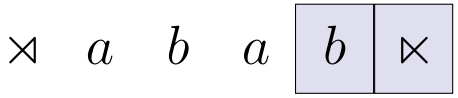




Strictly local acceptors

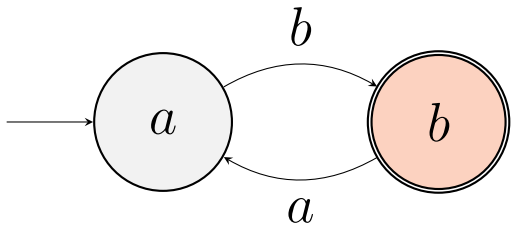
- Traversing a SL_kFSA is equivalent to scanning for k factors

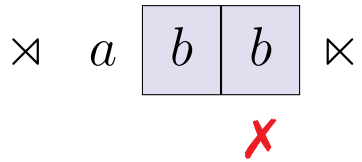




Strictly local acceptors

- Forbidden k-factors are expressed by missing transitions/accepting states





Strictly local acceptors

- SLFSAs describe exactly the SL languages
- Thus, they capture the **same** concept of locality as SL grammars, but in a different way

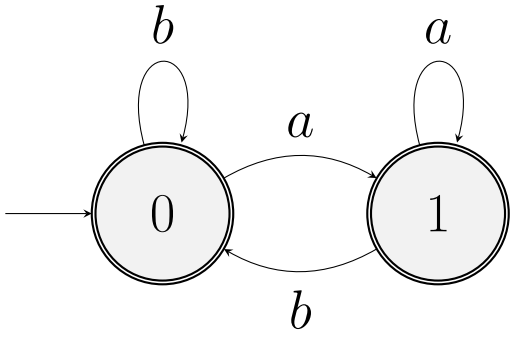
Learning with strictly local acceptors

Learning with strictly local acceptors

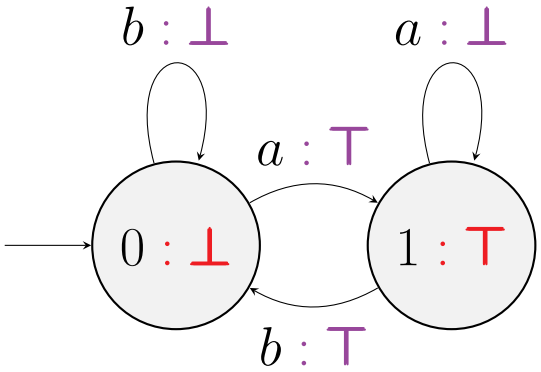
- Finite-state automata are useful because they have a number of learning techniques (de la Higuera, 2010)

- We'll use a 'transition filling' of Heinz and Rogers (2013)

Learning with strictly local acceptors

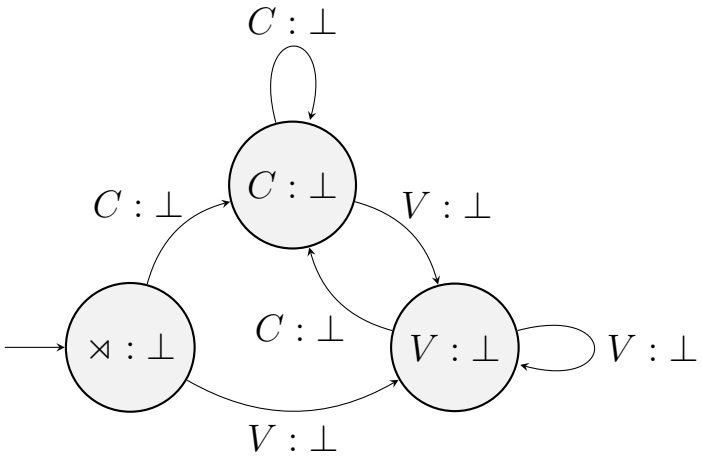


Learning with strictly local acceptors



- **output function** $Q \times \Sigma \to \{\top, \bot\}$
- **ending function** $Q \to \{\top, \bot\}$

Learning with strictly local acceptors

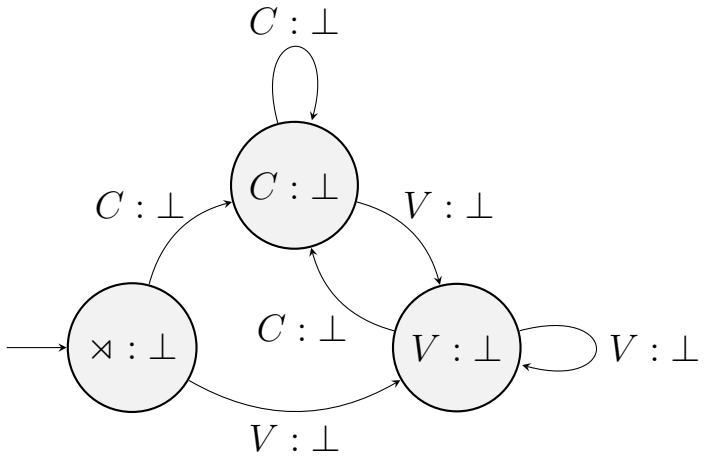


Learning procedure:

- Start with 'empty' SL_kFSA
- Change $\bot$ transitions to $\top$ when traversed by input data

Learning with strictly local acceptors

data
$0\ CV$

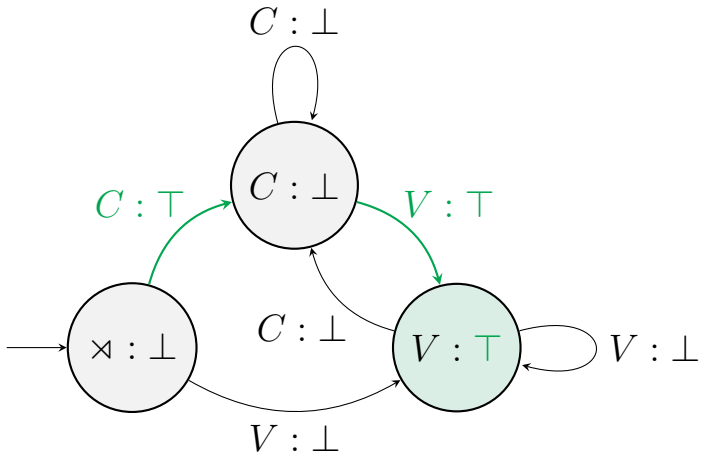


Learning procedure:

- Start with 'empty' SL_kFSA
- Change $\bot$ transitions to $\top$ when traversed by input data

Learning with strictly local acceptors

data
$0\ CV$

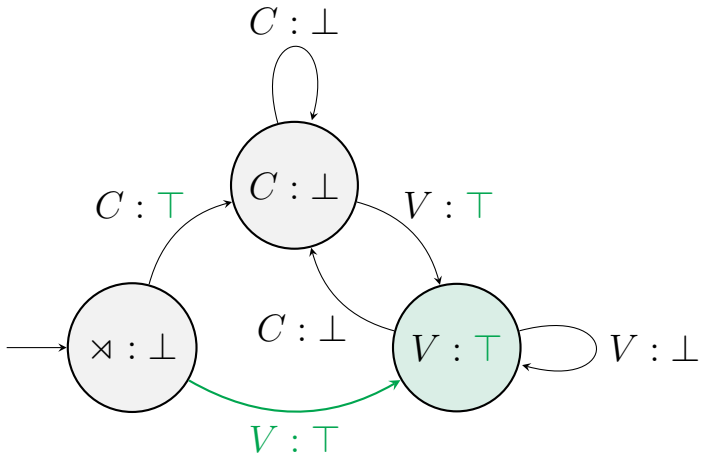


Learning procedure:

- Start with 'empty' SL_kFSA
- Change $\bot$ transitions to $\top$ when traversed by input data

Learning with strictly local acceptors

	data
0	CV
1	V

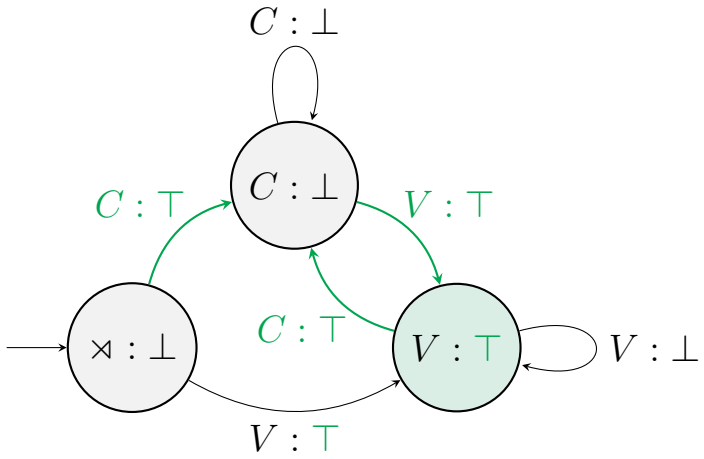


Learning procedure:

- Start with 'empty' SL_kFSA
- Change $\bot$ transitions to $\top$ when traversed by input data

Learning with strictly local acceptors

	data
0	CV
1	V
2	$CVCV$

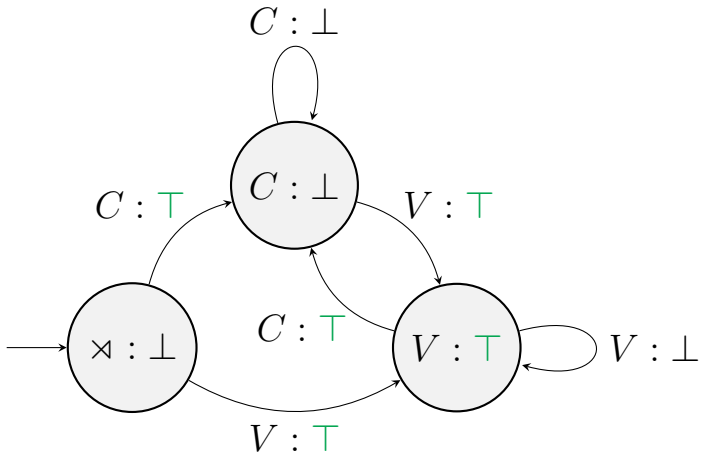


Learning procedure:

- Start with 'empty' SL_kFSA
- Change $\bot$ transitions to $\top$ when traversed by input data

Learning with strictly local acceptors

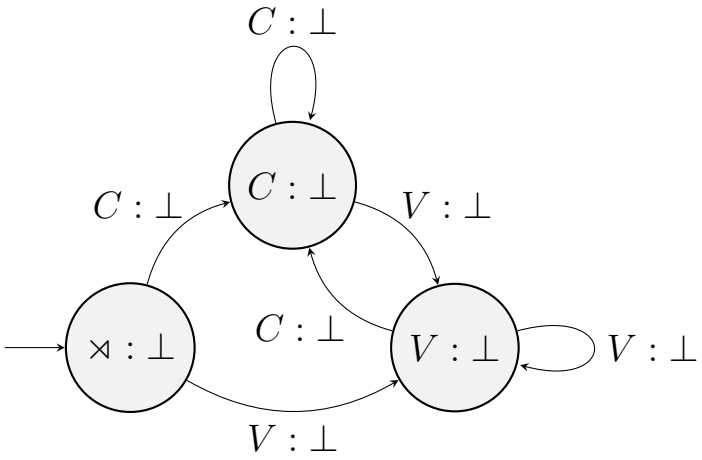
	data
0	CV
1	V
2	$CVCV$



Learning procedure:

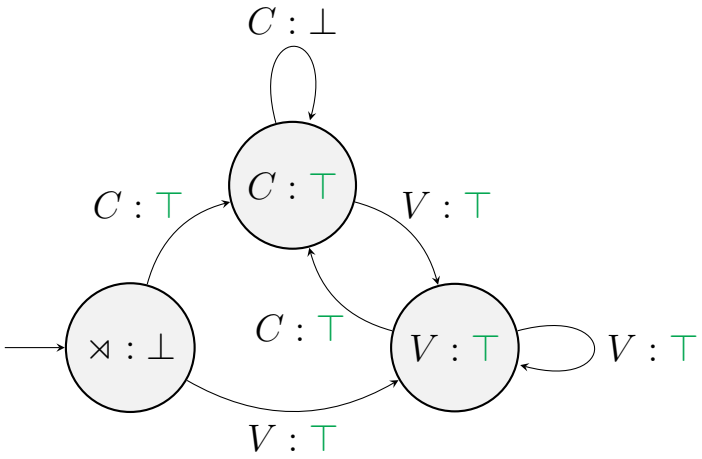
- Start with 'empty' SL_kFSA
- Change $\bot$ transitions to $\top$ when traversed by input data

Learning with strictly local acceptors



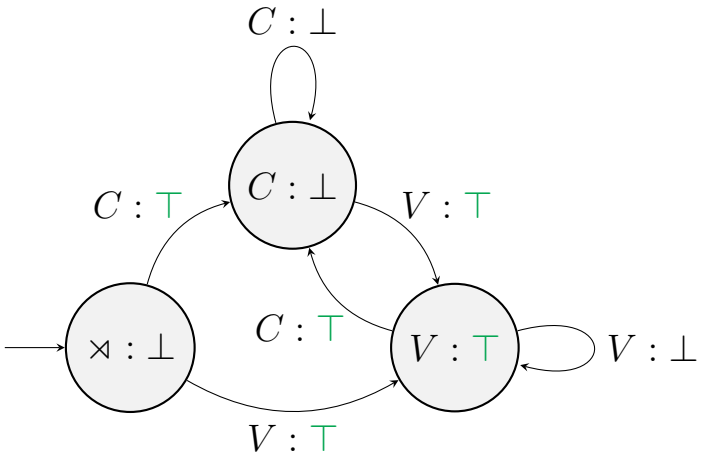
- **Any** SL_2 **language** can be described by varying $\{\top, \bot\}$ on this structure

Learning with strictly local acceptors



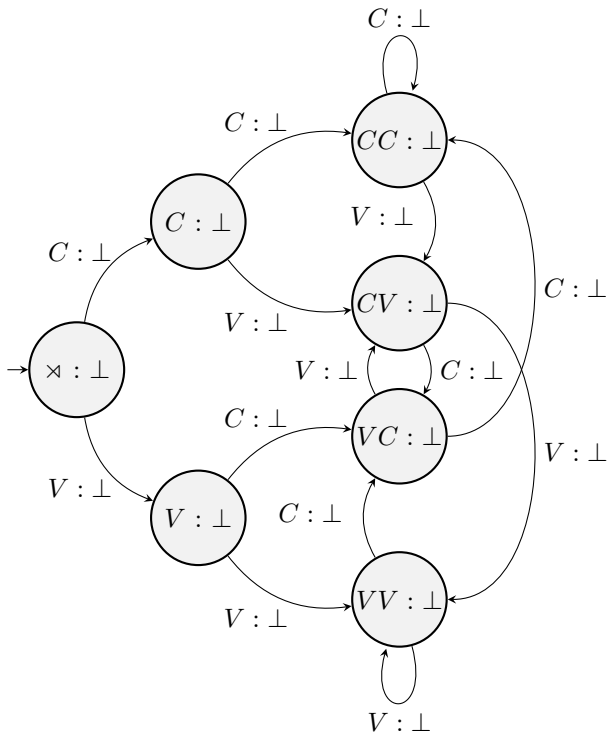
- **Any** SL_2 **language** can be described by varying $\{\top, \bot\}$ on this structure

Learning with strictly local acceptors



- **Any** SL_2 **language** can be described by varying $\{\top, \bot\}$ on this structure

Learning with strictly local acceptors



- **Any** SL_3 **language** can be described by this structure

Learning with strictly local acceptors

- This procedure **ILPD-learns any** SL_k **language for a given** k
- It is distinct, yet based on the **same notion** of locality

Input strictly local functions

Input strictly local functions

- Generative phonology is primarily interested in **maps**

$$/\text{kam-pa}/ \rightarrow [\text{kamba}]$$

/kam-pa/
b C → [+voi] / N __
[kamba]

	/kam-pa/	FAITH	*NC̬	ID(voi)
	[kampa]		*!	
	[kama]	*!		
☞	[kamba]			*

Input strictly local functions

- Maps are **(functional) relations**

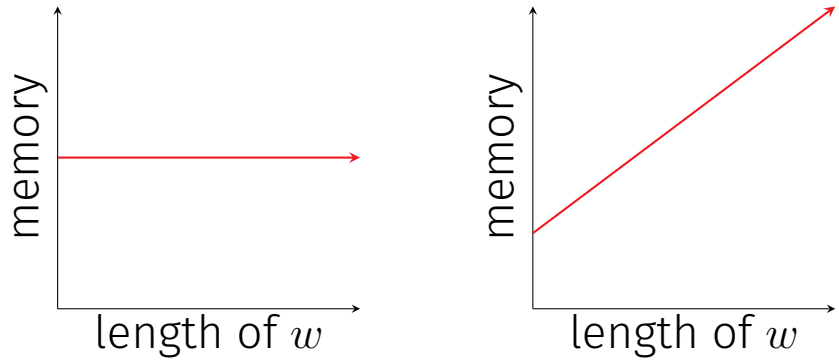
  /NC̥/ $\rightarrow$ [NC̬]

  $\{(\text{an}, \text{an}), (\text{anda}, \text{anda}), (\text{anta}, \text{anda}), (\text{lalalalampa}, \text{lalalalamba}), \ldots\}$

- We can study classes of relations like we studied classes of formal languages

Input strictly local functions

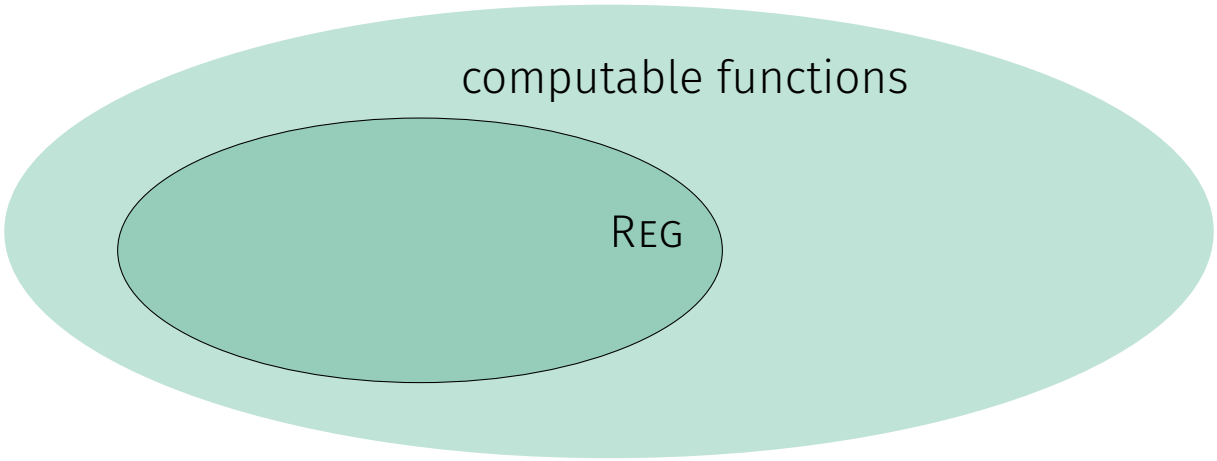
- Johnson (1972); Kaplan and Kay (1994): phonological maps are **regular**



regular **non-regular**

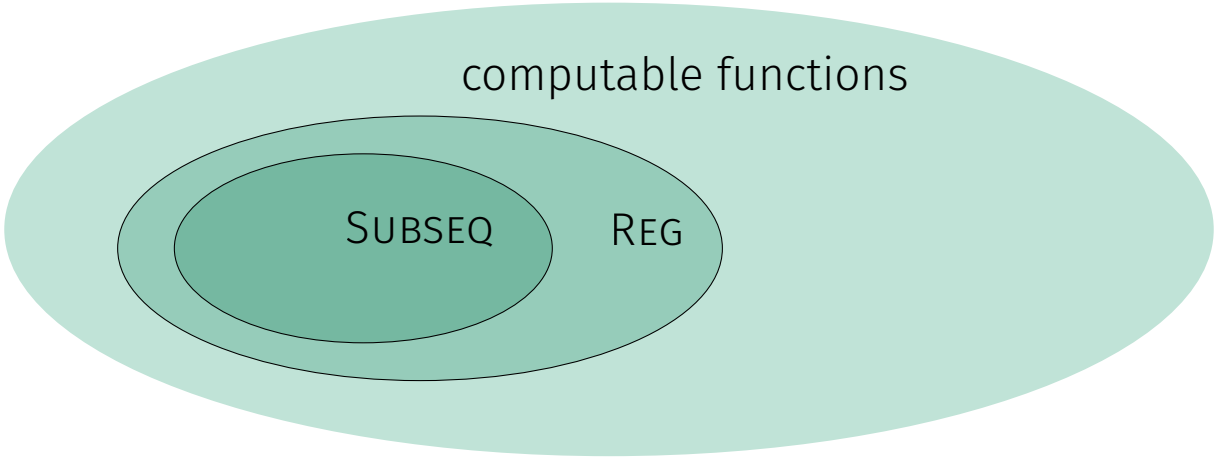
- **Regular functions $\neq$ regular languages!**

Input strictly local functions



- How do we extend strict locality to functions?

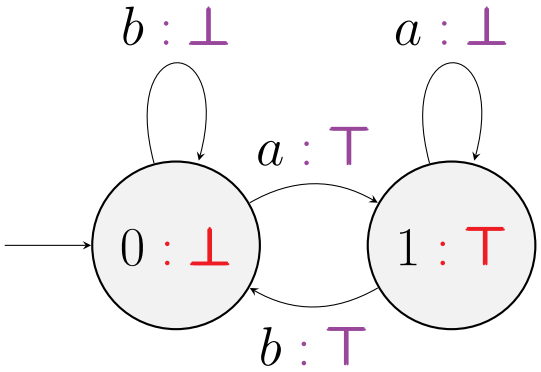
Input strictly local functions



- How do we extend strict locality to functions?
- Phonological maps are **subsequential**...

(Mohri, 1997; Heinz and Lai, 2013, et seq.)

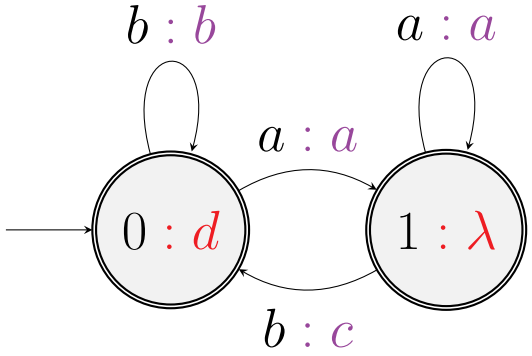
Subsequential transducers



Deterministic acceptor:

- **output function** $Q \times \Sigma \to \{\top, \bot\}$
- **ending function** $Q \to \{\top, \bot\}$

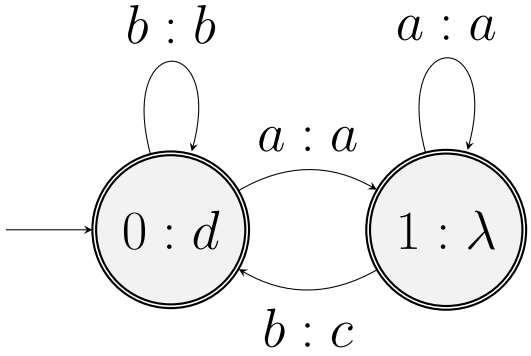
Subsequential transducers



Subsequential transducer:

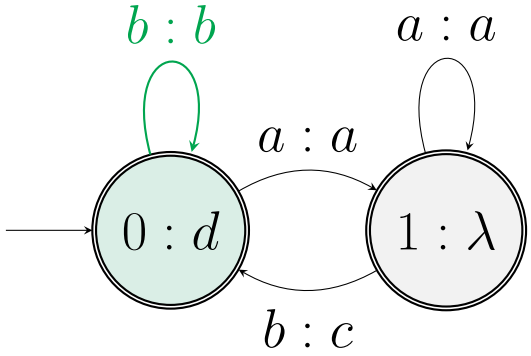
- **output function** $Q \times \Sigma \to \Gamma^*$
- **ending function** $Q \to \Gamma^*$

Subsequential transducers



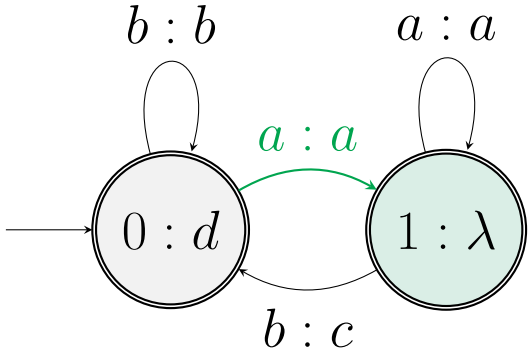
$b\ a\ b\ b$

Subsequential transducers



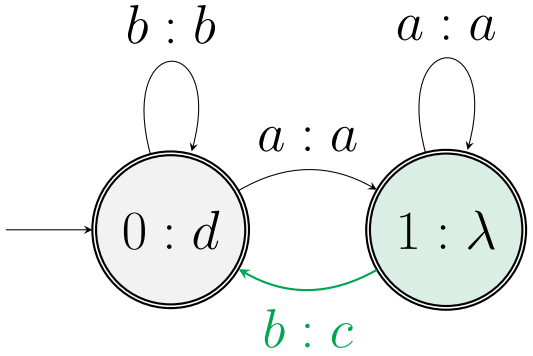
$$\begin{array}{ccccccc} & b & & a & b & b & \\ 0 & \rightarrow & 0 & & & & \\ & b & & & & & \end{array}$$

Subsequential transducers



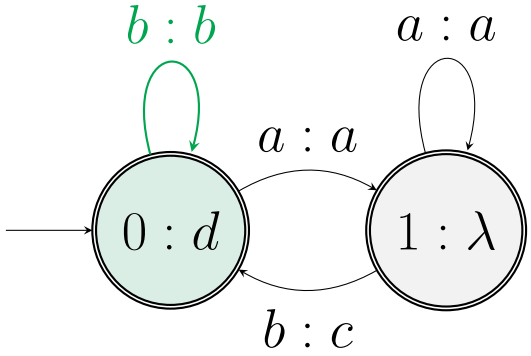
$$\begin{array}{ccccccc} & b & & a & & b & b \\ 0 & \rightarrow & 0 & \rightarrow & 1 & & \\ & b & & a & & & \end{array}$$

Subsequential transducers



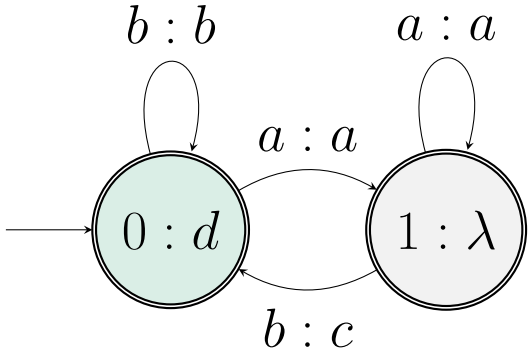
$$\begin{array}{ccccccccc} & & b & & a & & b & & b \\ 0 & \rightarrow & 0 & \rightarrow & 0 & \rightarrow & 1 & \rightarrow & 0 \\ & & b & & a & & c & & \end{array}$$

Subsequential transducers



$$\begin{array}{ccccccccc} & b & & a & & b & & b & \\ 0 & \to & 0 & \to & 1 & \to & 0 & \to & 0 \\ & b & & a & & c & & b & \end{array}$$

Subsequential transducers



$$\begin{array}{ccccccccc} & b & & a & & b & & b & \\ 0 & \rightarrow & 0 & \rightarrow & 1 & \rightarrow & 0 & \rightarrow & 0 \\ & b & & a & & c & & b & d \end{array}$$

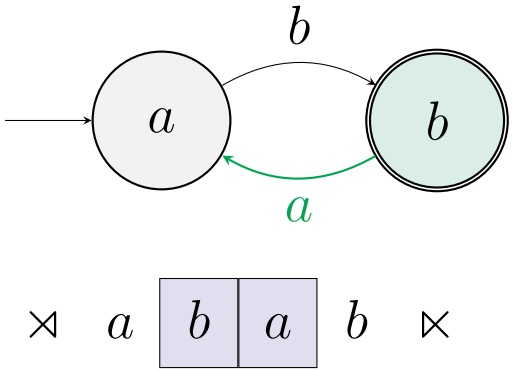
Subsequential transducers

Let’s do some examples...

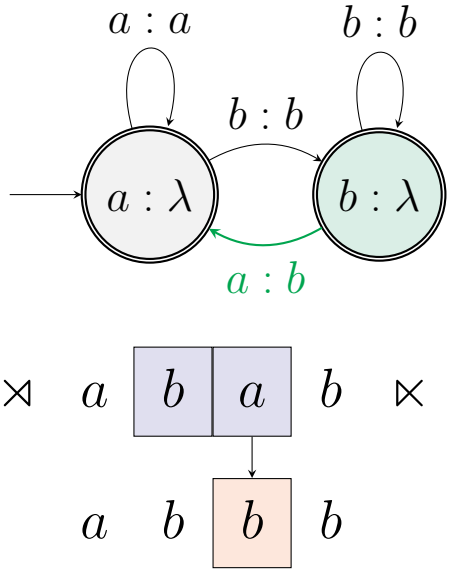
Input strictly local functions

- ISL transducers are SFSTs whose states represent $k-1$ suffixes. (Chandlee, 2014; Chandlee and Heinz, 2018)

SL acceptor:



ISL transducer:



Input strictly local functions

- 94% of the processes in P-Base (Mielke, 2004) are ISL (Chandlee and Heinz, 2018).

- Others are **output strictly local** (Chandlee, 2014) or are non-local, but subsequential (Luo, 2017; Payne, 2017)

Review

- **SL acceptors** exactly capture the SL notion of locality
- Learning with SL acceptors takes advantage of their **shared state structure**
- The **ISL functions** extend this structure from languages to functions