A Generalized Algorithm for Learning Positive and Negative Grammars with Unconventional String Models

Sarah Brogden Payne
sarah.payne@stonybrook.edu
paynesa.github.io

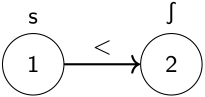
SCiL, UC Irvine
June 27, 2024

Overview

- Previous work: learning formal languages with **symbolic representations** (Heinz, 2010b; Heinz et al., 2012, i.a.)

Overview

- Previous work: learning formal languages with **symbolic representations** (Heinz, 2010b; Heinz et al., 2012, i.a.)
    - Grammars as collections of **forbidden** or **allowed** combinations

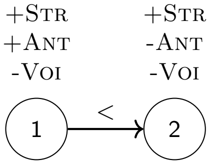


Overview

- Previous work: learning formal languages with **symbolic representations** (Heinz, 2010b; Heinz et al., 2012, i.a.)
  - Grammars as collections of **forbidden** or **allowed** combinations
- Chandlee et al. (2019): learn with **feature-based representations**

Overview

- Previous work: learning formal languages with **symbolic representations** (Heinz, 2010b; Heinz et al., 2012, i.a.)
  - Grammars as collections of **forbidden** or **allowed** combinations
- Chandlee et al. (2019): learn with **feature-based representations**
  - Grammars as collections of **forbidden** combinations



Overview

- Previous work: learning formal languages with **symbolic representations** (Heinz, 2010b; Heinz et al., 2012, i.a.)
  - Grammars as collections of **forbidden** or **allowed** combinations
- Chandlee et al. (2019): learn with **feature-based representations**
  - Grammars as collections of **forbidden** combinations
  - What about collections of **allowed** combinations?

Overview

- Previous work: learning formal languages with **symbolic representations** (Heinz, 2010b; Heinz et al., 2012, i.a.)
  - Grammars as collections of **forbidden** or **allowed** combinations
- Chandlee et al. (2019): learn with **feature-based representations**
  - Grammars as collections of **forbidden** combinations
  - What about collections of **allowed** combinations?
- **This talk:** algorithm for learning grammars as collections of **allowed** or **forbidden** feature-based combinations in a **unified** way

Example: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)

- Subsequences such as [s...s] that agree in $\pm$ANTERIOR are allowed
- Subsequences such as [s...ʃ] which disagree are banned

✓ [hasxintilawas] ✗ [hasxintilawaʃ]

(Hansson, 2010)

Example: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)

- Subsequences such as [s...s] that agree in $\pm$ANTERIOR are allowed
- Subsequences such as [s...ʃ] which disagree are banned

✓ [hasxintilawas] ✗ [hasxintilawaʃ]

(Hansson, 2010)

Negative Grammar (G^{-})

$[+\textsc{Ant}][\text{-}\textsc{Ant}] \in G^{-} \Rightarrow$ sʃt $\notin L(G^{-})$

Since $[+\textsc{Ant}][\text{-}\textsc{Ant}]$ covers [sʃ]

Example: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)

- Subsequences such as [s...s] that agree in $\pm$Anterior are allowed
- Subsequences such as [s...ʃ] which disagree are banned

✓ [hasxintilawas] ✗ [hasxintilawaʃ]

(Hansson, 2010)

Negative Grammar (G^{-})

$[+\textsc{Ant}][\text{-}\textsc{Ant}] \in G^{-} \Rightarrow$ sʃt $\notin L(G^{-})$

Since $[+\textsc{Ant}][\text{-}\textsc{Ant}]$ covers [sʃ]

Positive Grammar (G^{+})

$[+\textsc{Str}][\text{-}\textsc{Str}],[+\textsc{Ant}][\text{-}\textsc{Ant}] \in G^{+}$
$\Rightarrow$ sʃt $\in L(G^{+})$

Since $[+\textsc{Ant}][\text{-}\textsc{Ant}]$ covers [sʃ] and $[+\textsc{Str}][\text{-}\textsc{Str}]$ covers [ʃt]

Why Positive Grammars?

Psycholinguistic Motivation

Computational Motivation

Why Positive Grammars?

Psycholinguistic Motivation

- Child may construct **positive phonological** and **phonotactic** grammars (Belth, 2023; Payne, 2023, i.a.)

Computational Motivation

Why Positive Grammars?

Psycholinguistic Motivation

- Child may construct **positive phonological** and **phonotactic** grammars (Belth, 2023; Payne, 2023, i.a.)
- Evidence for positive **syntactic** and **morphological** grammars (Marcus et al., 1992; Yang, 2016; Belth et al., 2021; Li and Schuler, 2023, i.a.)

Computational Motivation

Why Positive Grammars?

Psycholinguistic Motivation

- Child may construct **positive phonological** and **phonotactic** grammars (Belth, 2023; Payne, 2023, i.a.)
- Evidence for positive **syntactic** and **morphological** grammars (Marcus et al., 1992; Yang, 2016; Belth et al., 2021; Li and Schuler, 2023, i.a.)

Computational Motivation

- **Post-hoc conversion** between positive & negative grammars is straightforward for symbolic models (Heinz, 2010b)

Why Positive Grammars?

Psycholinguistic Motivation

- Child may construct **positive phonological** and **phonotactic** grammars (Belth, 2023; Payne, 2023, i.a.)
- Evidence for positive **syntactic** and **morphological** grammars (Marcus et al., 1992; Yang, 2016; Belth et al., 2021; Li and Schuler, 2023, i.a.)

Computational Motivation

- **Post-hoc conversion** between positive & negative grammars is straightforward for symbolic models (Heinz, 2010b)
- Post-hoc conversion is **exponentially more costly** for models that use features

Why Positive Grammars?

Psycholinguistic Motivation

- Child may construct **positive phonological** and **phonotactic** grammars (Belth, 2023; Payne, 2023, i.a.)
- Evidence for positive **syntactic** and **morphological** grammars (Marcus et al., 1992; Yang, 2016; Belth et al., 2021; Li and Schuler, 2023, i.a.)

Computational Motivation

- **Post-hoc conversion** between positive & negative grammars is straightforward for symbolic models (Heinz, 2010b)
- Post-hoc conversion is **exponentially more costly** for models that use features
- Grammar polarity has implications for the **learning trajectory**

Why Positive Grammars?

Psycholinguistic Motivation

- Child may construct **positive phonological** and **phonotactic** grammars (Belth, 2023; Payne, 2023, i.a.)
- Evidence for positive **syntactic** and **morphological** grammars (Marcus et al., 1992; Yang, 2016; Belth et al., 2021; Li and Schuler, 2023, i.a.)

Computational Motivation

- **Post-hoc conversion** between positive & negative grammars is straightforward for symbolic models (Heinz, 2010b)
- Post-hoc conversion is **exponentially more costly** for models that use features
- Grammar polarity has implications for the **learning trajectory**

By **fixing** k — the size of the learned substructures — we can straightforwardly adapt the algorithm of Chandlee et al. (2019) to learn the **most general positive and negative grammars** over **feature-based models**

Table of Contents

Table of Contents

Finite Model Theory

Model Signature: a set of relations $R = \{R_1, R_2, ..., R_n\}$

- Each R_i is an m_i-ary relation

Finite Model Theory

Model Signature: a set of relations $R = \{R_1, R_2, ..., R_n\}$

- Each R_i is an m_i-ary relation

R-Structure: a tuple of elements $S = \langle D; R_1, R_2, ..., R_n \rangle$

- D, **the domain**, is a finite set of elements
- Each R_i is a subset of D^{m_i}

Finite Model Theory

Model Signature: a set of relations $R = \{R_1, R_2, ..., R_n\}$

- Each R_i is an m_i-ary relation

R-Structure: a tuple of elements $S = \langle D; R_1, R_2, ..., R_n \rangle$

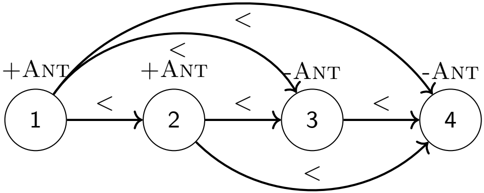
- D, **the domain**, is a finite set of elements
- Each R_i is a subset of D^{m_i}

Size $|S|$ of an R-structure = cardinality of its domain

Precedence and Successor Models

Precedence Model: $M^{<}(w) := \langle D^w; <, [R^w_\sigma]_{\sigma\in\Sigma}\rangle$

- $D^w = \{1, ..., |w|\}$ is the **domain** of positions in w
- $<:= \{(i,j) \in D^w \times D^w \mid i < j\}$ is the **general precedence** relation

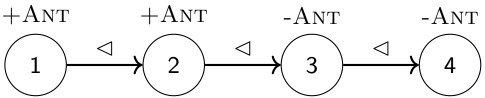


(Büchi, 1960; McNaughton and Papert, 1971; Rogers et al., 2013)

Precedence and Successor Models

Successor Model: $M^{\lhd}(w) \coloneqq \langle D^w; \lhd, [R^w_\sigma]_{\sigma \in \Sigma} \rangle$

- $D^w = \{1, ..., |w|\}$ is the **domain** of positions in w
- $\lhd \coloneqq \{(i, i+1) \in D^w \times D^w\}$ is the **successor** relation



(Büchi, 1960; McNaughton and Papert, 1971; Rogers et al., 2013)

Table of Contents

Subfactors & Maxfactors

An R-structure A is a **subfactor** of an R-structure B (notated $A \sqsubseteq B$) if:

Subfactors & Maxfactors

An R-structure A is a **subfactor** of an R-structure B (notated $A \sqsubseteq B$) if:

1. There is a mapping between D^A and some subset of D^B

Subfactors & Maxfactors

An R-structure A is a **subfactor** of an R-structure B (notated $A \sqsubseteq B$) if:

1. There is a mapping between D^A and some subset of D^B
2. All relations that hold in A also hold over the corresponding elements in B

Subfactors & Maxfactors

An R-structure A is a **subfactor** of an R-structure B (notated $A \sqsubseteq B$) if:

1. There is a mapping between D^A and some subset of D^B
2. All relations that hold in A also hold over the corresponding elements in B

An R-structure A is a **maxfactor** of an R-structure B (notated $A \leq B$) if **1 and 2 are satisfied** and:

Subfactors & Maxfactors

An R-structure A is a **subfactor** of an R-structure B (notated $A \sqsubseteq B$) if:

1. There is a mapping between D^A and some subset of D^B
2. All relations that hold in A also hold over the corresponding elements in B

An R-structure A is a **maxfactor** of an R-structure B (notated $A \leq B$) if **1 and 2 are satisfied** and:

3. All relations that hold in B also hold over the corresponding elements in A

Subfactors & Maxfactors

An R-structure A is a **subfactor** of an R-structure B (notated $A \sqsubseteq B$) if:

1. There is a mapping between D^A and some subset of D^B
2. All relations that hold in A also hold over the corresponding elements in B

An R-structure A is a **maxfactor** of an R-structure B (notated $A \leq B$) if **1 and 2 are satisfied** and:

3. All relations that hold in B also hold over the corresponding elements in A

Subfactor: Unidirectional

Subfactors & Maxfactors

An R-structure A is a **subfactor** of an R-structure B (notated $A \sqsubseteq B$) if:

1. There is a mapping between D^A and some subset of D^B
2. All relations that hold in A also hold over the corresponding elements in B

An R-structure A is a **maxfactor** of an R-structure B (notated $A \leq B$) if **1 and 2 are satisfied** and:

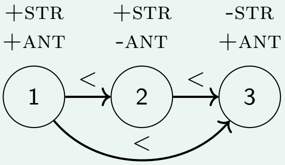
3. All relations that hold in B also hold over the corresponding elements in A

Subfactor: Unidirectional

Maxfactor: Bidirectional

Subfactors vs. Maxfactors

Examples (Samala Sibilant Harmony)

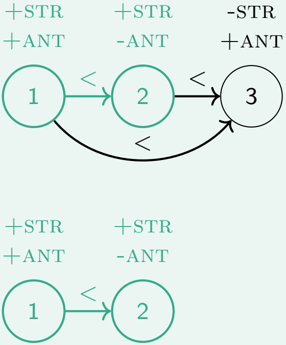


Subfactor: Unidirectional

Maxfactor: Bidirectional

Subfactors vs. Maxfactors

Examples (Samala Sibilant Harmony)

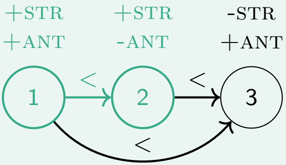


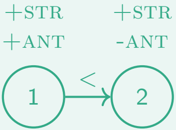
Subfactor: Unidirectional

Maxfactor: Bidirectional

Subfactors vs. Maxfactors

Examples (Samala Sibilant Harmony)



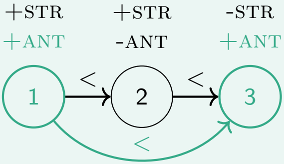


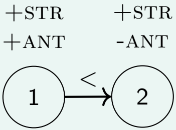
✓Subfactor: Unidirectional

✓Maxfactor: Bidirectional

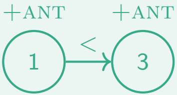
Subfactors vs. Maxfactors

Examples (Samala Sibilant Harmony)





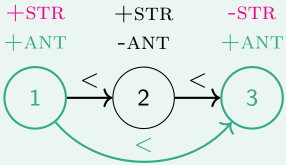
Subfactor: Unidirectional

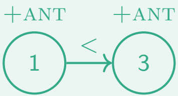


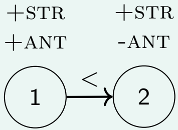
Maxfactor: Bidirectional

Subfactors vs. Maxfactors

Examples (Samala Sibilant Harmony)





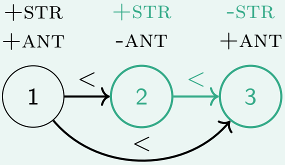


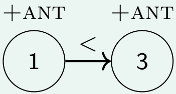
✓Subfactor: Unidirectional

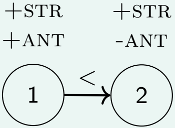
✗Maxfactor: Bidirectional

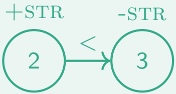
Subfactors vs. Maxfactors

Examples (Samala Sibilant Harmony)







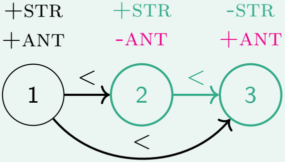


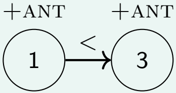
Subfactor: Unidirectional

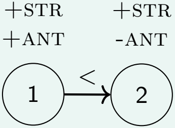
Maxfactor: Bidirectional

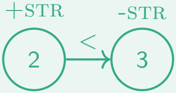
Subfactors vs. Maxfactors

Examples (Samala Sibilant Harmony)









✓Subfactor: Unidirectional

✗Maxfactor: Bidirectional

k-Subfactors and k-Maxfactors

Definition: k-Subfactors

If $A \sqsubseteq B$ and $|A| = k$, then A is a ***k*-subfactor** of B

Definition: k-Maxfactors

If $A \leq B$ and $|A| = k$, then A is ***k*-maxfactor** of B

k-Subfactors and k-Maxfactors

Definition: k-Subfactors

If $A \sqsubseteq B$ and $|A| = k$, then A is a **k-subfactor** of B

Let the set of k-subfactors of an R-structure B be given by:

$$\textsc{Sfac}_k(B) := \{A \mid A \sqsubseteq B, |A| = k\}$$

Definition: k-Maxfactors

If $A \leq B$ and $|A| = k$, then A is **k-maxfactor** of B

Let the set of k-maxfactors of B be given by:

$$\textsc{Mfac}_k(B) := \{A \mid A \leq B, |A| = k\}$$

Table of Contents

Positive vs. Negative Grammars

Grammar G = finite set of k-subfactors

Language defined by G depends on its **interpretation**:

Positive vs. Negative Grammars

Grammar G = finite set of k-subfactors
Language defined by G depends on its **interpretation**:

Negative Grammar (G^-)

Elements of G^- are forbidden, and strings in $L(G^-)$ contain no forbidden subfactors

Positive vs. Negative Grammars

Grammar G = finite set of k-subfactors
Language defined by G depends on its **interpretation**:

Negative Grammar (G^-)

Elements of G^- are forbidden, and strings in $L(G^-)$ contain no forbidden subfactors

Positive Grammar (G^+)

Elements of G^+ are permissible, and strings in $L(G^+)$ are those which are *tiled* by these elements

Positive Grammars: Tiling

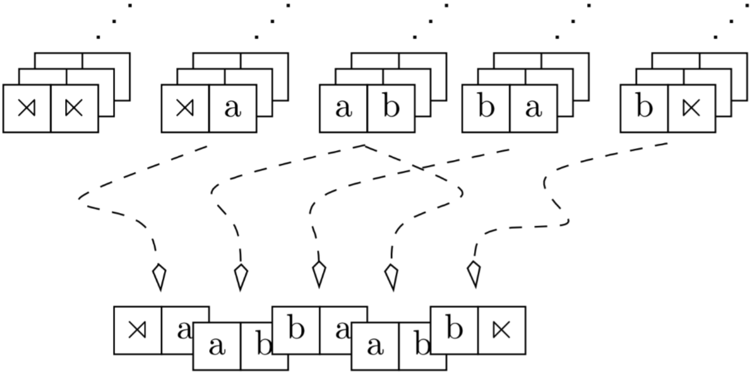


Figure courtesy of Rogers and Heinz (2014)

Languages of Positive vs. Negative Grammars

Negative Grammar

The language $L(G^-)$ of G^- is given by:

$$L(G^-) = \{w \in \Sigma^* \mid (\forall S \in \textsc{MFac}_k(M, w)) \; [\textsc{SFac}_k(S) \cap G^- = \emptyset]\}$$

or equivalently by:

$$L(G^-) = \{w \in \Sigma^* \mid (\nexists S \in \textsc{MFac}_k(M, w)) \; [\textsc{SFac}_k(S) \cap G^- \neq \emptyset]\}$$

Positive Grammar

The language $L(G^+)$ of G^+ is given by:

$$L(G^+) = \{w \in \Sigma^* \mid (\forall S \in \textsc{MFac}_k(M, w)) \; [\textsc{SFac}_k(S) \cap G^+ \neq \emptyset]\}$$

or equivalently by:

$$L(G^+) = \{w \in \Sigma^* \mid (\nexists S \in \textsc{MFac}_k(M, w)) \; [\textsc{SFac}_k(S) \cap G^+ = \emptyset]\}$$

Languages of Positive vs. Negative Grammars

Negative Grammar

The language $L(G^-)$ of G^- is given by:

$$L(G^-) = \{w \in \Sigma^* \mid (\forall S \in \textsc{Mfac}_k(M, w)) \\ [\textsc{Sfac}_k(S) \cap G^- = \emptyset]\}$$

or equivalently by:

$$L(G^-) = \{w \in \Sigma^* \mid (\nexists S \in \textsc{Mfac}_k(M, w)) \\ [\textsc{Sfac}_k(S) \cap G^- \neq \emptyset]\}$$

Positive Grammar

The language $L(G^+)$ of G^+ is given by:

$$L(G^+) = \{w \in \Sigma^* \mid (\forall S \in \textsc{Mfac}_k(M, w)) \\ [\textsc{Sfac}_k(S) \cap G^+ \neq \emptyset]\}$$

or equivalently by:

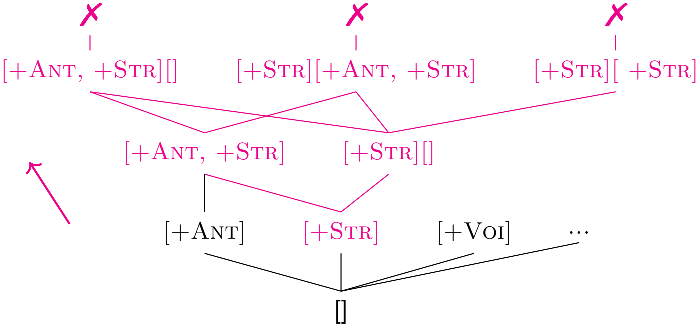
$$L(G^+) = \{w \in \Sigma^* \mid (\nexists S \in \textsc{Mfac}_k(M, w)) \\ [\textsc{Sfac}_k(S) \cap G^+ = \emptyset]\}$$

	$\in G$	$\notin G$
$\forall$	**Positive Grammar**	**Negative Grammar**
$\exists$	**Negative Grammar**	**Positive Grammar**

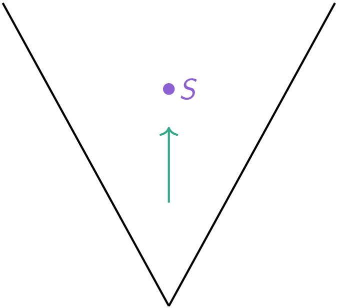
Table of Contents

Previous Work

Crucial insight of **Chandlee et al. (2019)**: **grammatical entailment**

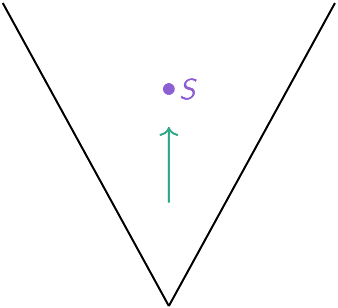


Chandlee et al. (2019) Algorithm



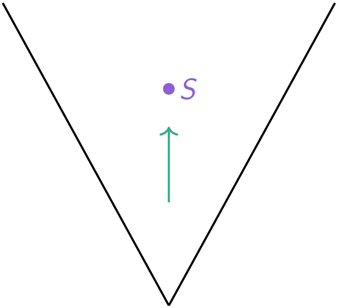
- **Bottom-up** traversal

Chandlee et al. (2019) Algorithm



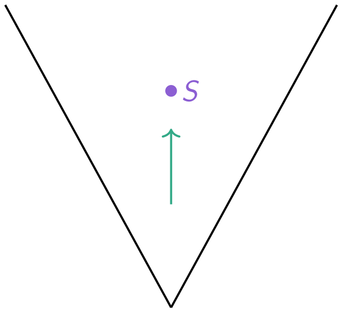
- **Bottom-up** traversal
- For a given subfactor S, check whether $S \sqsubseteq x$ for any x in the data D

Chandlee et al. (2019) Algorithm



- **Bottom-up** traversal
- For a given subfactor S, check whether $S \sqsubseteq x$ for any x in the data D
  - If not, **posit a constraint:** $S \in G^{-}$

Chandlee et al. (2019) Algorithm



- **Bottom-up** traversal
- For a given subfactor S, check whether $S \sqsubseteq x$ for any x in the data D
  - If not, **posit a constraint:** $S \in G^-$
  - If so, **cannot posit a constraint**
    Add the **least superfactors** of S to the queue to be considered next

Least Superfactors

Least Superfactors of S ($\texttt{NextSupFact}(S)$) are the superfactors of S that differ minimally from S

Least Superfactors

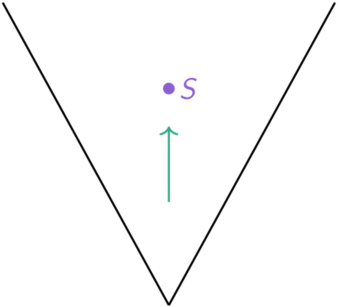
Least Superfactors of S ($\texttt{NextSupFact}(S)$) are the superfactors of S that differ minimally from S

Examples (Next superfactors of $[+\textsc{Ant}]$)

If $S = [+\textsc{Ant}]$ and the only features available are $\pm\textsc{Ant}$, $\pm\textsc{Voi}$ and $\pm\textsc{Str}$:

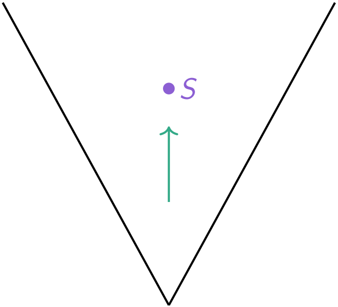
$$\begin{aligned}\texttt{NextSupFact}(S) = \{&[+\textsc{Ant},\ \text{-}\textsc{Str}], [+\textsc{Ant},\ +\textsc{Str}]\\ &[+\textsc{Ant},\ \text{-}\textsc{Voi}], [+\textsc{Ant},\ +\textsc{Voi}]\}\end{aligned}$$

Intuition



For a **negative grammar**:

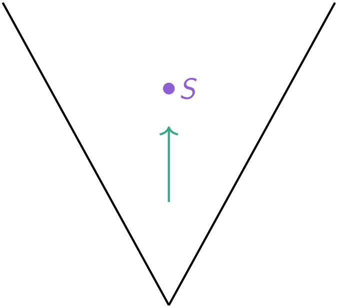
- Add S to G^- if $S \not\sqsubseteq x$ for any $x \in D$



For a **negative grammar**:

- Add S to G^- if $S \not\sqsubseteq x$ for any $x \in D$
- **Equivalent:** Given the set of all maxfactors that are superfactors of S, **none are attested** in D

Intuition

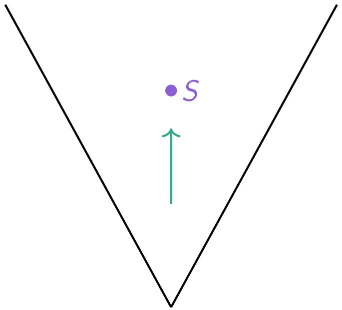


For a **negative grammar**:

- Add S to G^- if $S \not\sqsubseteq x$ for any $x \in D$
- **Equivalent:** Given the set of all maxfactors that are superfactors of S, **none are attested** in D

	$\in D$	$\notin D$
$\forall$	**Positive Grammar**	**Negative Grammar**
$\exists$	Negative Grammar	Positive Grammar

Intuition



For a **negative grammar**:

- Add S to G^- if $S \not\sqsubseteq x$ for any $x \in D$
- **Equivalent:** Given the set of all maxfactors that are superfactors of S, **none are attested** in D

	$\in D$	$\notin D$
$\forall$	**Positive Grammar**	**Negative Grammar**
$\exists$	Negative Grammar	Positive Grammar

For a **positive grammar**:

- Given the set of all maxfactors that are superfactors of S, **all are attested** in D

Extensions

Extensions of S ($\textsc{Ext}_k(S)$) are all k-maxfactors that are superfactors of S

Extensions

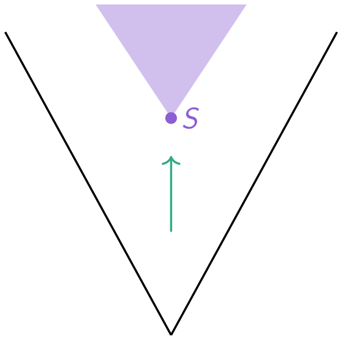
Extensions of S ($\textsc{Ext}_k(S)$) are all k-maxfactors that are superfactors of S

Examples (Extension of $[+\textsc{Ant}]$)

If $S = [+\textsc{Ant}]$ and the only features available are $\pm\textsc{Ant}$, $\pm\textsc{Voi}$ and $\pm\textsc{Str}$:

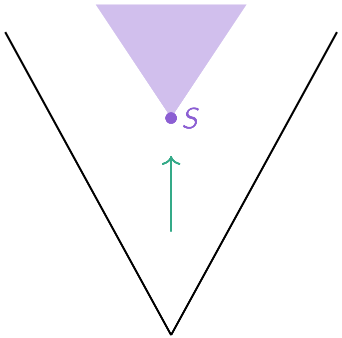
$$\begin{aligned}\textsc{Ext}_k(S) = \{&[+\textsc{Ant},\ \text{-}\textsc{Str},\ +\textsc{Voi}], [+\textsc{Ant},\ +\textsc{Str},\ +\textsc{Voi}]\\ &[+\textsc{Ant},\ \text{-}\textsc{Str},\ \text{-}\textsc{Voi}], [+\textsc{Ant},\ +\textsc{Str},\ \text{-}\textsc{Voi}]\}\end{aligned}$$

Learning Algorithm



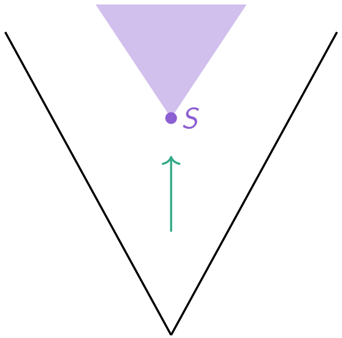
- **Bottom-up** traversal

Learning Algorithm



- **Bottom-up** traversal
- For a given subfactor S, check whether:

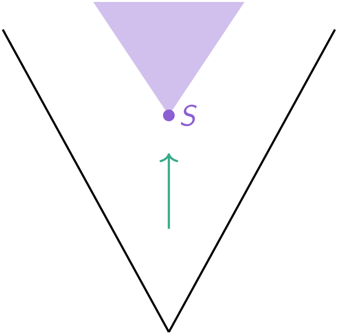
Learning Algorithm



- **Bottom-up** traversal
- For a given subfactor S, check whether:
  - G is **negative** and

$$(\forall s' \in \textsc{Ext}_k(S))[\nexists x \in D, s' \leq x])$$

Learning Algorithm



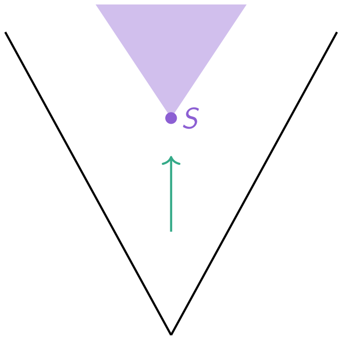
- **Bottom-up** traversal
- For a given subfactor S, check whether:
  - G is **negative** and

    $$(\forall s' \in \textsc{Ext}_k(S))[\nexists x \in D, s' \leq x])$$

  - G is **positive** and

    $$(\forall s' \in \textsc{Ext}_k(S))[\exists x \in D, s' \leq x])$$

Learning Algorithm



- **Bottom-up** traversal
- For a given subfactor S, check whether:
  - G is **negative** and

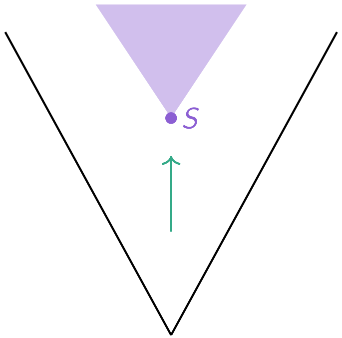
    $$(\forall s' \in \textsc{Ext}_k(S))[\nexists x \in D, s' \leq x])$$

  - G is **positive** and

    $$(\forall s' \in \textsc{Ext}_k(S))[\exists x \in D, s' \leq x])$$

- If either condition is met:
  - **Add S to G!**

Learning Algorithm



- **Bottom-up** traversal
- For a given subfactor S, check whether:
  - G is **negative** and

    $$(\forall s' \in \textsc{Ext}_k(S))[\nexists x \in D, s' \leq x])$$

  - G is **positive** and

    $$(\forall s' \in \textsc{Ext}_k(S))[\exists x \in D, s' \leq x])$$

- If either condition is met:
  - **Add S to G!**
- Otherwise:
  - Add the **minimal superfactors** of S to the queue to be considered next

Learning Guarantees

Fix Σ, model M, positive integer k, and polarity p. For any language $L \in \mathscr{L}^p(M, k)$ and for any finite sample $D \subseteq L$, return a grammar G^p such that:

Learning Guarantees

Fix Σ, model M, positive integer k, and polarity p. For any language $L \in \mathscr{L}^p(M, k)$ and for any finite sample $D \subseteq L$, return a grammar G^p such that:

1. G^p is **consistent**, that is, $D \subseteq L(G^p)$.

Learning Guarantees

Fix Σ, model M, positive integer k, and polarity p. For any language $L \in \mathscr{L}^p(M, k)$ and for any finite sample $D \subseteq L$, return a grammar G^p such that:

1. G^p is **consistent**, that is, $D \subseteq L(G^p)$.
2. $L(G^p)$ is a **smallest language** in $\mathscr{L}^p(M, k)$ which covers D, so that for all $L \in \mathscr{L}^p(M, k)$ where $D \subseteq L$, we have $L(G^p) \subseteq L$.

Learning Guarantees

Fix Σ, model M, positive integer k, and polarity p. For any language $L \in \mathscr{L}^p(M, k)$ and for any finite sample $D \subseteq L$, return a grammar G^p such that:

1. G^p is **consistent**, that is, $D \subseteq L(G^p)$.
2. $L(G^p)$ is a **smallest language** in $\mathscr{L}^p(M, k)$ which covers D, so that for all $L \in \mathscr{L}^p(M, k)$ where $D \subseteq L$, we have $L(G^p) \subseteq L$.
3. G^p includes R-structures S that are restrictions of R-structures S' in other grammars G' that also satisfy (1) and (2). That is, for all G' satisfying (1) and (2) and for all $S' \in G'$, there exists some $S \in G^p$ such that $S \sqsubseteq S'$.

Table of Contents

Applying the Algorithm: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)

Simplifying Assumptions:

Applying the Algorithm: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)

Simplifying Assumptions:

- Two features: $\pm$ANT **(relevant)** and $\pm$VOI **(irrelevant)**

Applying the Algorithm: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)

Simplifying Assumptions:

- Two features: $\pm$ANT **(relevant)** and $\pm$VOI **(irrelevant)**
- $k = 2$

Applying the Algorithm: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)

Simplifying Assumptions:

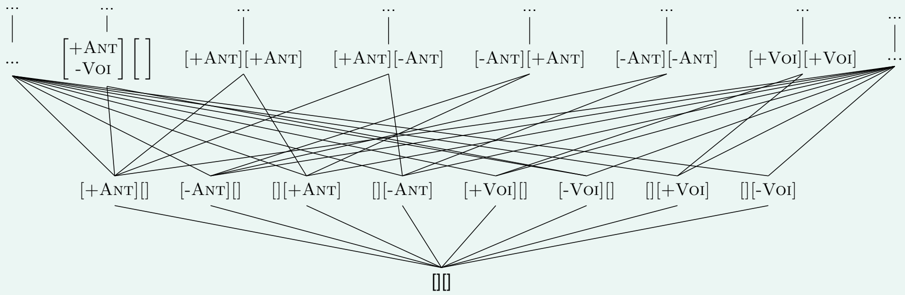
- Two features: $\pm$ANT **(relevant)** and $\pm$VOI **(irrelevant)**
- $k = 2$
- All licit subsequences are attested (cf. Heinz, 2010a)

Applying the Algorithm: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)

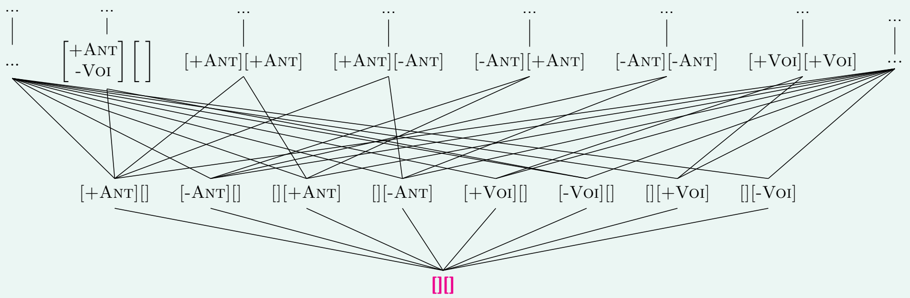
Simplifying Assumptions:

- Two features: $\pm$Ant **(relevant)** and $\pm$Voi **(irrelevant)**
- $k = 2$
- All licit subsequences are attested (cf. Heinz, 2010a)



Applying the Algorithm: Samala Sibilant Harmony

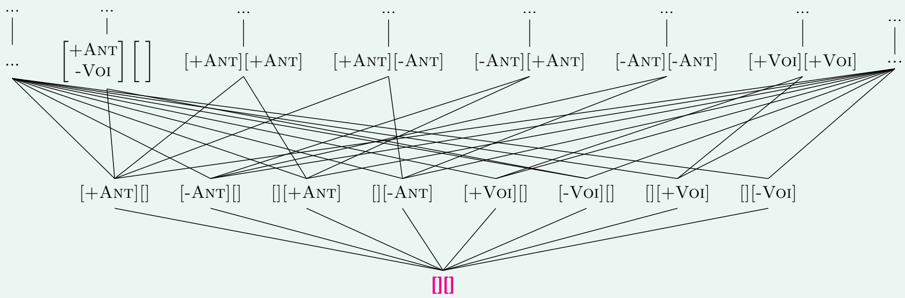
Examples (Samala Sibilant Harmony)



Negative Grammar

Positive Grammar

Examples (Samala Sibilant Harmony)



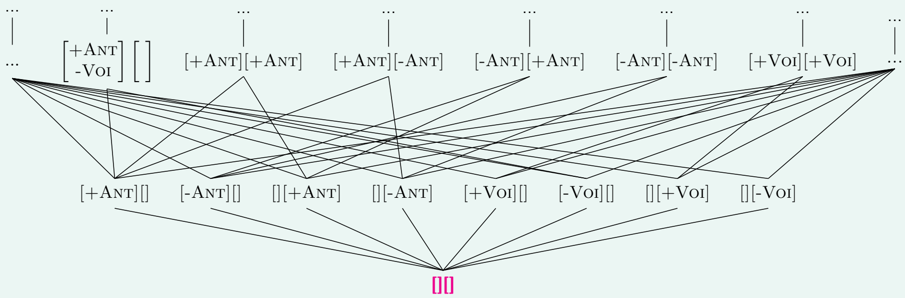
Negative Grammar

Is there any element in $\text{Ext}_k([][])$ which **is** a 2-maxfactor of some $x \in D$?

Positive Grammar

Applying the Algorithm: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)



Negative Grammar

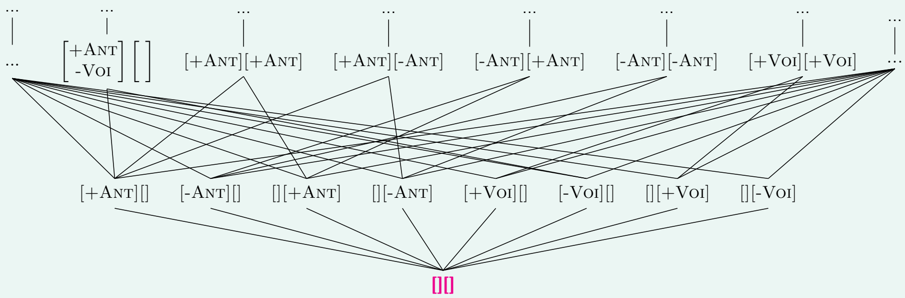
Is there any element in $\textsc{Ext}_k([][])$ which **is** a 2-maxfactor of some $x \in D$?

Positive Grammar

Is there any element in $\textsc{Ext}_k([][])$ which **is not** a 2-maxfactor of some $x \in D$?

Applying the Algorithm: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)



Negative Grammar

Is there any element in $\textsc{Ext}_k([][])$ which **is** a 2-maxfactor of some $x \in D$?

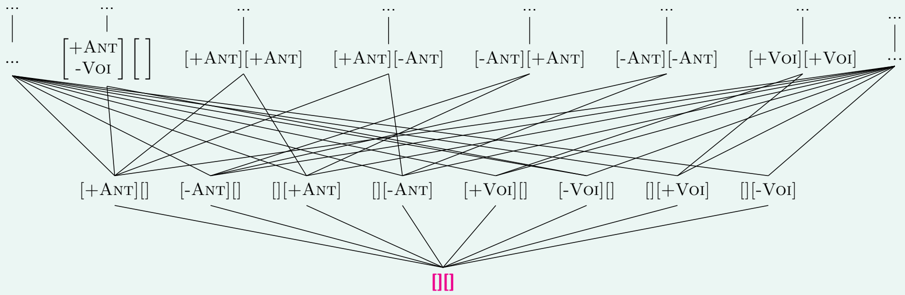
e.g. $[+\textsc{Voi}, +\textsc{Ant}][+\textsc{Voi}, +\textsc{Ant}] \in \textsc{Ext}_k([][])$, but [z...z] is **licit and attested**.

Positive Grammar

Is there any element in $\textsc{Ext}_k([][])$ which **is not** a 2-maxfactor of some $x \in D$?

Applying the Algorithm: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)



Negative Grammar

Is there any element in $\textsc{Ext}_k$([][]) which **is** a 2-maxfactor of some $x \in D$?

e.g. [+Voi, +Ant][+Voi, +Ant] $\in \textsc{Ext}_k$([][]), but [z...z] is **licit and attested**.

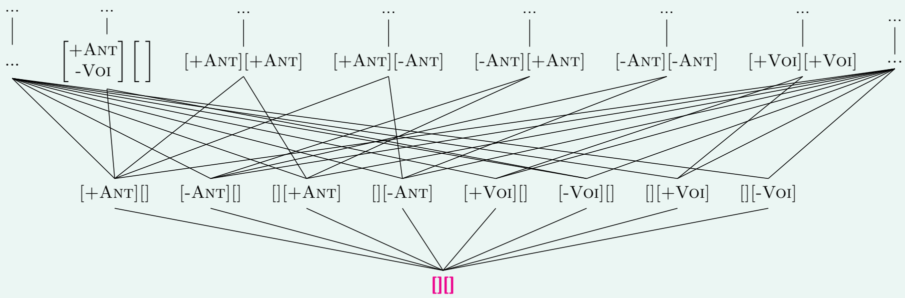
Positive Grammar

Is there any element in $\textsc{Ext}_k$([][]) which **is not** a 2-maxfactor of some $x \in D$?

e.g. [+Voi, +Ant][+Voi, -Ant] $\in \textsc{Ext}_k$([][]), but [z...ʒ] is **illicit and unattested**.

Applying the Algorithm: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)



Negative Grammar

Is there any element in $\textsc{Ext}_k([][])$ which **is** a 2-maxfactor of some $x \in D$?

e.g. $[+\textsc{Voi}, +\textsc{Ant}][+\textsc{Voi}, +\textsc{Ant}] \in \textsc{Ext}_k([][])$, but [z...z] is **licit and attested**.

Positive Grammar

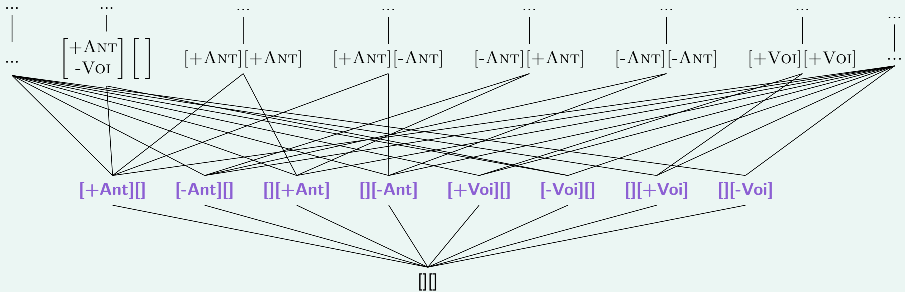
Is there any element in $\textsc{Ext}_k([][])$ which **is not** a 2-maxfactor of some $x \in D$?

e.g. $[+\textsc{Voi}, +\textsc{Ant}][+\textsc{Voi}, -\textsc{Ant}] \in \textsc{Ext}_k([][])$, but [z...ʒ] is **illicit and unattested**.

Keep searching!

Applying the Algorithm: Samala Sibilant Harmony

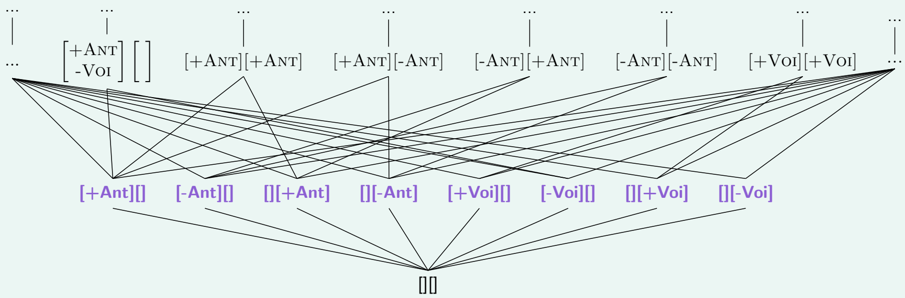
Examples (Samala Sibilant Harmony)



- Extract the **least superfactors** of [][] and consider each of them

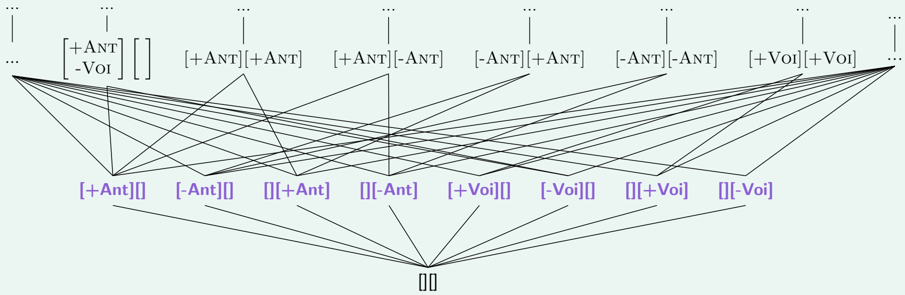
Applying the Algorithm: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)



- Extract the **least superfactors** of [][] and consider each of them
- Still not specified enough:

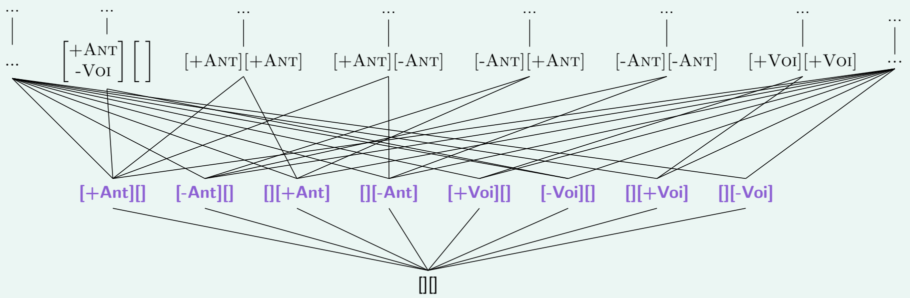
Examples (Samala Sibilant Harmony)



- Extract the **least superfactors** of [][] and consider each of them
- Still not specified enough:
  - Any subfactor with $\pm$Ant specified in one position has licit and illicit maxfactors in its extension

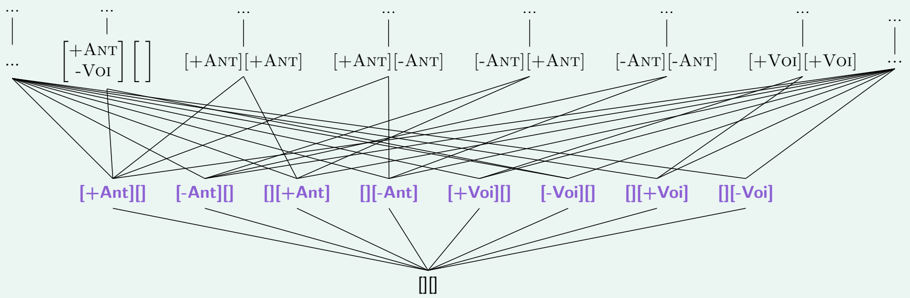
Applying the Algorithm: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)



- Extract the **least superfactors** of [][] and consider each of them
- Still not specified enough:
  - Any subfactor with ±Ant specified in one position has licit and illicit maxfactors in its extension
    [+Ant][] $\sqsubseteq$ [+Ant][+Ant] but [+Ant][] $\sqsubseteq$ [+Ant][-Ant]

Examples (Samala Sibilant Harmony)

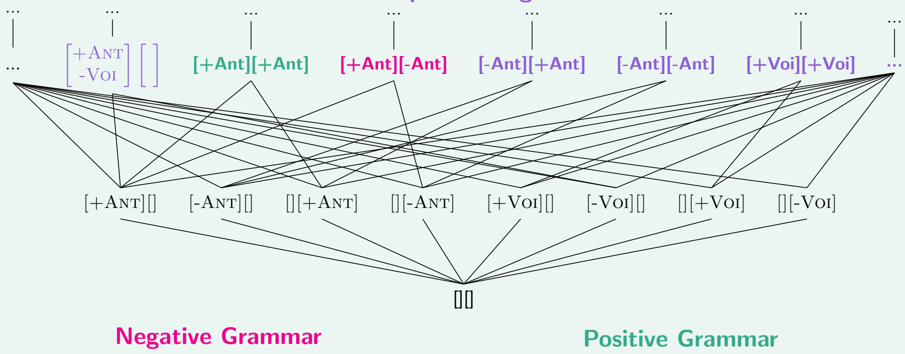


- Extract the **least superfactors** of [][] and consider each of them
- Still not specified enough:
  - Any subfactor with $\pm$ANT specified in one position has licit and illicit maxfactors in its extension
    [+ANT][] $\sqsubseteq$ [+ANT][+ANT] but [+ANT][] $\sqsubseteq$ [+ANT][-ANT]
  - $\pm$VOI has no bearing on licitness

Applying the Algorithm: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)

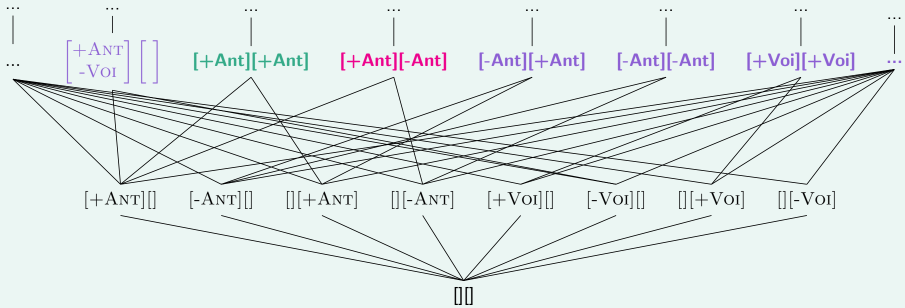
Keep searching!



Applying the Algorithm: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)

Keep searching!



Negative Grammar

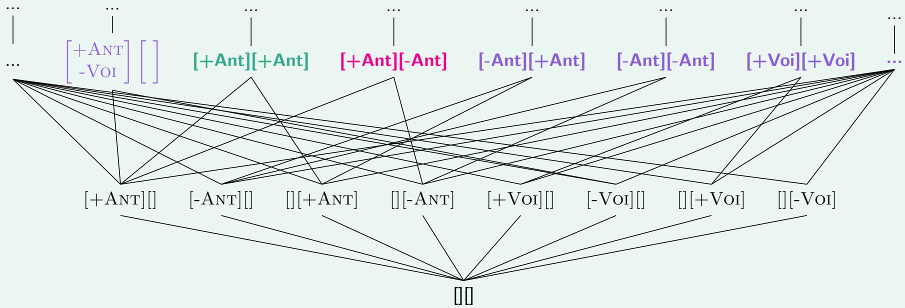
Is there any element in $\textsc{Ext}_k([+\textsc{Ant}][-\textsc{Ant}])$ which **is** a 2-maxfactor of some $x \in D$? **No!**

Positive Grammar

Applying the Algorithm: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)

Keep searching!



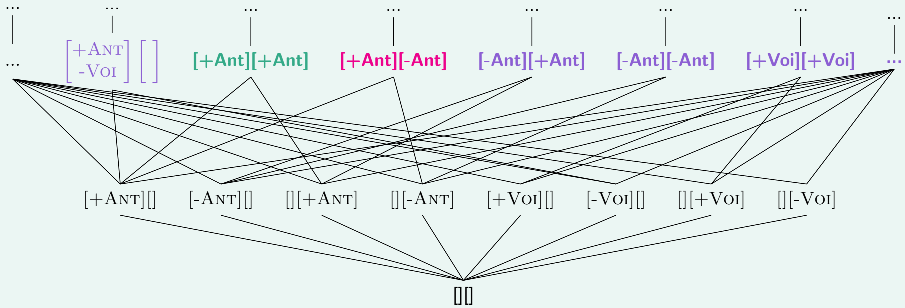
Negative Grammar
Is there any element in $\textsc{Ext}_k([+\textsc{Ant}][-\textsc{Ant}])$ which **is** a 2-maxfactor of some $x \in D$? **No!**

Positive Grammar
Is there any element in $\textsc{Ext}_k([+\textsc{Ant}][+\textsc{Ant}])$ which **is not** a 2-maxfactor of some $x \in D$? **No!**

Applying the Algorithm: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)

Keep searching!



Negative Grammar

Is there any element in $\textsc{Ext}_k$([+ANT][-ANT]) which **is** a 2-maxfactor of some $x \in D$? **No!**

[+ANT][-ANT] is added to G^-

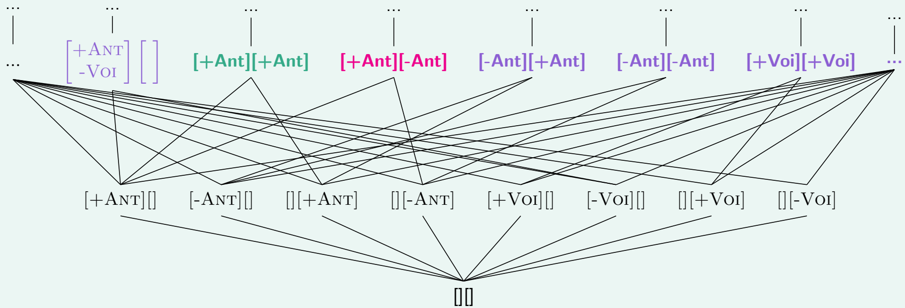
Positive Grammar

Is there any element in $\textsc{Ext}_k$([+ANT][+ANT]) which **is not** a 2-maxfactor of some $x \in D$? **No!**

[+ANT][+ANT] is added to G^+

Applying the Algorithm: Samala Sibilant Harmony

Examples (Samala Sibilant Harmony)



Negative Grammar

We may later reach [+Ant][-Ant, +Voi] but we won't consider it.

[+Ant][-Ant] being banned entails [+Ant][-Ant, +Voi] being banned

Positive Grammar

We may later reach [+Ant][+Ant, +Voi] but we won't consider it.

[+Ant][+Ant] being allowed entails [+Ant][+Ant, +Voi] being allowed

Summary & Discussion

- If we fix the size k of subfactors in the grammar, the algorithm of Chandlee et al. (2019) can be adapted to learn **positive** and **negative** grammars in a **unified** way

Summary & Discussion

- If we fix the size k of subfactors in the grammar, the algorithm of Chandlee et al. (2019) can be adapted to learn **positive** and **negative** grammars in a **unified** way
- Enriched representations of feature-based string models allow us to provably find the **most general subfactors**

Summary & Discussion

- If we fix the size k of subfactors in the grammar, the algorithm of Chandlee et al. (2019) can be adapted to learn **positive** and **negative** grammars in a **unified** way
- Enriched representations of feature-based string models allow us to provably find the **most general subfactors**
- Implications of **grammar polarity**:

Summary & Discussion

- If we fix the size k of subfactors in the grammar, the algorithm of Chandlee et al. (2019) can be adapted to learn **positive** and **negative** grammars in a **unified** way
- Enriched representations of feature-based string models allow us to provably find the **most general subfactors**
- Implications of **grammar polarity**:
  - As G^+ grows, $L(G^+)$ **grows**

Summary & Discussion

- If we fix the size k of subfactors in the grammar, the algorithm of Chandlee et al. (2019) can be adapted to learn **positive** and **negative** grammars in a **unified** way
- Enriched representations of feature-based string models allow us to provably find the **most general subfactors**
- Implications of **grammar polarity**:
  - As G^+ grows, $L(G^+)$ **grows**
  - As G^- grows, $L(G^-)$ **shrinks**

Summary & Discussion

- If we fix the size k of subfactors in the grammar, the algorithm of Chandlee et al. (2019) can be adapted to learn **positive** and **negative** grammars in a **unified** way
- Enriched representations of feature-based string models allow us to provably find the **most general subfactors**
- Implications of **grammar polarity**:
  - As G^+ grows, $L(G^+)$ **grows**
  - As G^- grows, $L(G^-)$ **shrinks**
  - Initially, G^+ allows **nothing**, while G^- allows **everything**

Future Work

- **Implementing** the generalized algorithm and applying it to corpus data

Future Work

- **Implementing** the generalized algorithm and applying it to corpus data
- Approaches to dealing with **noise** and **sparsity** in the data

Future Work

- **Implementing** the generalized algorithm and applying it to corpus data
- Approaches to dealing with **noise** and **sparsity** in the data
- Applying the algorithm to learning **non-sequential** representations

Future Work

- **Implementing** the generalized algorithm and applying it to corpus data
- Approaches to dealing with **noise** and **sparsity** in the data
- Applying the algorithm to learning **non-sequential** representations
- Learning mixed grammars containing both **banned** and **allowed** subfactors

Future Work

- **Implementing** the generalized algorithm and applying it to corpus data
- Approaches to dealing with **noise** and **sparsity** in the data
- Applying the algorithm to learning **non-sequential** representations
- Learning mixed grammars containing both **banned** and **allowed** subfactors
- Comparison of **learning trajectories** of **positive** & **negative** grammars

Future Work

- **Implementing** the generalized algorithm and applying it to corpus data
- Approaches to dealing with **noise** and **sparsity** in the data
- Applying the algorithm to learning **non-sequential** representations
- Learning mixed grammars containing both **banned** and **allowed** subfactors
- Comparison of **learning trajectories** of **positive** & **negative** grammars
  - Within a single search of the hypothesis space

Future Work

- **Implementing** the generalized algorithm and applying it to corpus data
- Approaches to dealing with **noise** and **sparsity** in the data
- Applying the algorithm to learning **non-sequential** representations
- Learning mixed grammars containing both **banned** and **allowed** subfactors
- Comparison of **learning trajectories** of **positive** & **negative** grammars
    - Within a single search of the hypothesis space
    - When applied to incrementally larger data sets as a proxy for incremental learning

Thank you!

I am grateful to Jeff Heinz, Thomas Graf, Jon Rawski, Logan Swanson, and the SCiL reviewers for discussion.

This work was supported by the Institute for Advanced Computational Science Graduate Research Fellowship and the National Science Foundation Graduate Research Fellowship Program.

References I

Caleb Belth. 2023. *Towards an Algorithmic Account of Phonological Rules and Representations*. Ph.D. thesis, University of Michigan.

Caleb Belth, Sarah Payne, Deniz Beser, Jordan Kodner, and Charles Yang. 2021. The greedy and recursive search for morphological productivity. *Proceedings of the 43rd annual meeting of the Cognitive Science Society*, 43:2869–2875.

J Richard Büchi. 1960. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6).

Jane Chandlee, Remi Eyraud, Jeffrey Heinz, Adam Jardine, and Jonathan Rawski. 2019. Learning with partially ordered representations. In *Proceedings of the 16th Meeting on the Mathematics of Language*, pages 91–101, Toronto, Canada. Association for Computational Linguistics.

Gunnar Ólafur Hansson. 2010. *Consonant harmony: Long-distance interactions in phonology*, volume 145. University of California Press.

Jeffrey Heinz. 2010a. Learning long-distance phonotactics. *Linguistic Inquiry*, 41(4):623–661.

Jeffrey Heinz. 2010b. String extension learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 897–906, Uppsala, Sweden. Association for Computational Linguistics.

References II

Jeffrey Heinz, Anna Kasprzik, and Timo Kötzing. 2012. Learning in the limit with lattice-structured hypothesis spaces. *Theoretical Computer Science*, 457:111–127.

Daoxin Li and Kathryn D Schuler. 2023. Acquiring recursive structures through distributional learning. *Language Acquisition*, pages 1–14.

Gary F Marcus, Steven Pinker, Michael Ullman, Michelle Hollander, T John Rosen, Fei Xu, and Harald Clahsen. 1992. Overregularization in language acquisition. *Monographs of the society for research in child development*, pages i–178.

Robert McNaughton and Seymour A Papert. 1971. *Counter-Free Automata (MIT research monograph no. 65)*. The MIT Press.

Sarah Payne. 2023. Marginal sequences are licit but unproductive. Poster presented at the 2023 Annual Meeting of Phonology.

James Rogers and Jeffrey Heinz. 2014. Model theoretic phonology. In *Workshop slides in the 26th European Summer School in Logic, Language and Information*.

James Rogers, Jeffrey Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. 2013. Cognitive and sub-regular complexity. In *Formal Grammar: 17th and 18th International Conferences, Revised Selected Papers*, pages 90–108. Springer.

References III

Kristina Strother-Garcia, Jeffrey Heinz, and Hyun Jin Hwangbo. 2016. Using model theory for grammatical inference: a case study from phonology. In *Proceedings of The 13th International Conference on Grammatical Inference*, pages 66–78.

Mai H Vu, Ashkan Zehfroosh, Kristina Strother-Garcia, Michael Sebok, Jeffrey Heinz, and Herbert G Tanner. 2018. Statistical relational learning with unconventional string models. *Frontiers in Robotics and AI*, 5:76.

Charles Yang. 2016. *The price of linguistic productivity: How children learn to break the rules of language*. MIT press.

Restrictions

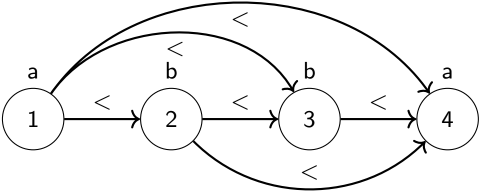
Definition: Restriction

An R-structure A is a **restriction** of an R-structure B if $D^A \subseteq D^B$ and for each m-ary relation R_i in the model signature:

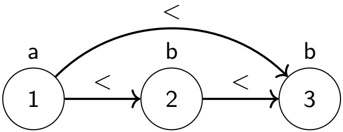
$$R_i^A = \{(x_1, ..., x_m) \in R_i^B \mid x_1, ..., x_m \in D^A\} \quad (1)$$

Intuition: identify a subset D^A of the domain of B and retain only those relations in B whose elements are wholly within D^A

R-structure B



R-structure A



$D^A = \{1, 2, 3\} \subset D^B = \{1, 2, 3, 4\}$

Subfactor

Definition: Subfactor

An R-structure A is a **subfactor** of an R-structure B (notated $A \sqsubseteq B$) if there exists a restriction B' of B and a bijection h such that for all $R_i \in R$, if $R_i(x_1, ..., x_m)$ holds in A, then $R_i(h(x_1), ..., h(x_m))$ holds in B'.

Intuition: A is a subfactor of B if there is a mapping between D^A and some subset of D^B and all relations that hold in A also hold over the corresponding elements in B

Maxfactor

Definition: Maxfactor

An R-structure A is a **maxfactor** of an R-structure B (notated $A \leq B$) iff $A \sqsubseteq B$ and for each m-ary relation R_i, whenever $R_i(x_1, ..., x_m)$ holds in B, $R_i(h^{-1}(x_1), ..., h^{-1}(x_m))$ holds in A.

Intuition: A is a maxfactor of B if $A \sqsubseteq B$ and and all relations that hold in B also hold over the corresponding elements in A

Extensions

Extensions of a Subfactor

The extensions of a subfactor S are defined as follows:

$$\begin{aligned}\textsc{Ext}_k(S) = &\{A \in \textsc{Sfac}_k(M, \Sigma^*) \mid \\ &S \sqsubseteq A \wedge (\nexists A')[|A'| = k \wedge A \sqsubseteq A']\}\end{aligned} \tag{2}$$

Intuition: extensions of S are all k-maxfactors that are superfactors of S.

Examples (Extension of $[+\textsc{Ant}]$)

If $S = [+\textsc{Ant}]$ and the only features available are $\pm\textsc{Ant}$, $\pm\textsc{Voi}$ and $\pm\textsc{Str}$:

$$\begin{aligned}\textsc{Ext}_k(S) = &\{[+\textsc{Ant},\ \text{-}\textsc{Str},\ +\textsc{Voi}], [+\textsc{Ant},\ +\textsc{Str},\ +\textsc{Voi}] \\ &[+\textsc{Ant},\ \text{-}\textsc{Str},\ \text{-}\textsc{Voi}], [+\textsc{Ant},\ +\textsc{Str},\ \text{-}\textsc{Voi}]\}\end{aligned}$$

Next Superfactor

Next Superfactor

We extract the more specific superfactors of S by calling `NextSupFact`(s) where `NextSupFact()` is defined as follows:

$$\begin{aligned}\texttt{NextSupFact}(S) = &\{A \in \textsc{SFac}_k(M, \Sigma^*) \mid \\ &S \sqsubseteq A \wedge (\nexists A')[S \sqsubseteq A' \sqsubseteq A]\}\end{aligned} \tag{3}$$

Intuition: `NextSupFact()` returns the *least* superfactors for S.

Examples (Next superfactors of $[+\textsc{Ant}]$)

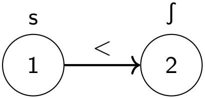
If $S = [+\textsc{Ant}]$ and the only features available are $\pm\textsc{Ant}$, $\pm\textsc{Voi}$ and $\pm\textsc{Str}$:

$$\begin{aligned}\texttt{NextSupFact}(S) = &\{[+\textsc{Ant},\ \text{-}\textsc{Str}], [+\textsc{Ant},\ +\textsc{Str}] \\ &[+\textsc{Ant},\ \text{-}\textsc{Voi}], [+\textsc{Ant},\ +\textsc{Voi}]\}\end{aligned}$$

Conventional vs. Unconventional String Models

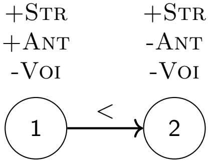
Conventional String Models

- **Mutually-exclusive unary relations** label each domain element with the single property of being some $\sigma \in \Sigma$
- **Segments** in phonological applications



Unconventional String Models

- **Non-exclusive unary relations** allow distinct alphabetic symbols to share properties
- **Features** in phonological applications



(Strother-Garcia et al., 2016; Vu et al., 2018)

Connectedness

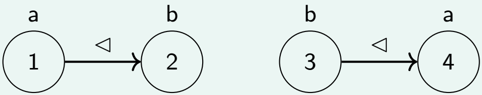
Connectedness

An R-structure $S = \langle D; R_1, R_2, ..., R_n \rangle$ is connected iff $(\forall x, y \in D)[(x, y) \in C^*]$, where C^* is defined as the symmetric transitive closure of:

$$\begin{aligned} C = &\{(x, y) \in D \times D \mid \\ &\quad \exists i \in \{1...n\}, \exists (x_1...x_m) \in R_i \\ &\quad \exists s, t \in \{1...m\}, x = x_s, y = x_t\} \end{aligned}$$

Intuition: domain elements x and y of S belong to C if they belong to some non-unary relation R_i in S

Examples (Disconnected R-Structure)



The Cost of Interdefinability

For symbolic models, negative & positive grammars are straightforwardly interdefinable:

$$G^+ = \Sigma^k \setminus G^- \quad G^- = \Sigma^k \setminus G^+$$

Two complications for feature-based models:

Number of k-Subfactors

- A model with n binary features defines $s \leq 2^n$ segments
- Segment-based model: no more than $(s)^k \leq (2^n)^k$ k-factors
- Under a feature-based model: $(3^n)^k$ possible k-subfactors since each feature can be positive, negative, or unspecified

Conversion Process

- For symbolic models, need to simply check whether some k-factor $f \in \Sigma^k$ is in G^-
  if $[\text{s}\ldots\text{ʃ}] \in G^-$ then $[\text{s}\ldots\text{ʃ}] \notin G^+$
- For feature-based models, a k-subfactor f should not be added to G^+ if $f \in G^-$, but also if $(\exists g \in G^-)[f \sqsubseteq g \vee g \sqsubseteq f]$.

Some Lemmas

Lemma 1: Maxfactor-Subfactor Containment

Let k be some positive integer and let M be some model of Σ^*. For any $w \in \Sigma^*$ and for any $F \in \textsc{Sfac}_k(M, w)$, we have that:

$$[\exists G \in \textsc{Mfac}_k(M, w)](F \sqsubseteq G)$$

Lemma 2: Union of Subfactors of Maxfactors

Let k be some positive integer and let M be some model of Σ^*. For any $w \in \Sigma^*$, we have that:

$$\bigcup_{S \in \textsc{Mfac}_k(M,w)} \textsc{Sfac}_k(S) = \textsc{Sfac}_k(M, w)$$