Python toolkit for subregular languages

Alëna Aksënova

Linguistics department @ SBU

IACS Research Day

IACS @ SBU

April 4, 2018

Introduction

Today, I will...

- talk about the subregular hierarchy
  very useful for linguistics and not only
- explain what is the `kist` toolkit
  provides different subregular tools
- update on what has been done so far
  learners, scanners, generators for several language classes

Outline

The side of linguistics

For linguistics, it is important to be able to formalize patterns and to study properties of the system.

The side of linguistics

For linguistics, it is important to be able to formalize patterns and to study properties of the system.

Among others, it can give us

- typological predictions;
- language models;
- cognitive insights.

The side of formal language theory

For formal language theory, it is important to know different types of patterns and their complexity.

The side of formal language theory

For formal language theory, it is important to know different types of patterns and their complexity.

Among others, it can give us

- formal properties of the patterns;
- generating devices they require;
- cognitive applicability.

Subregular languages

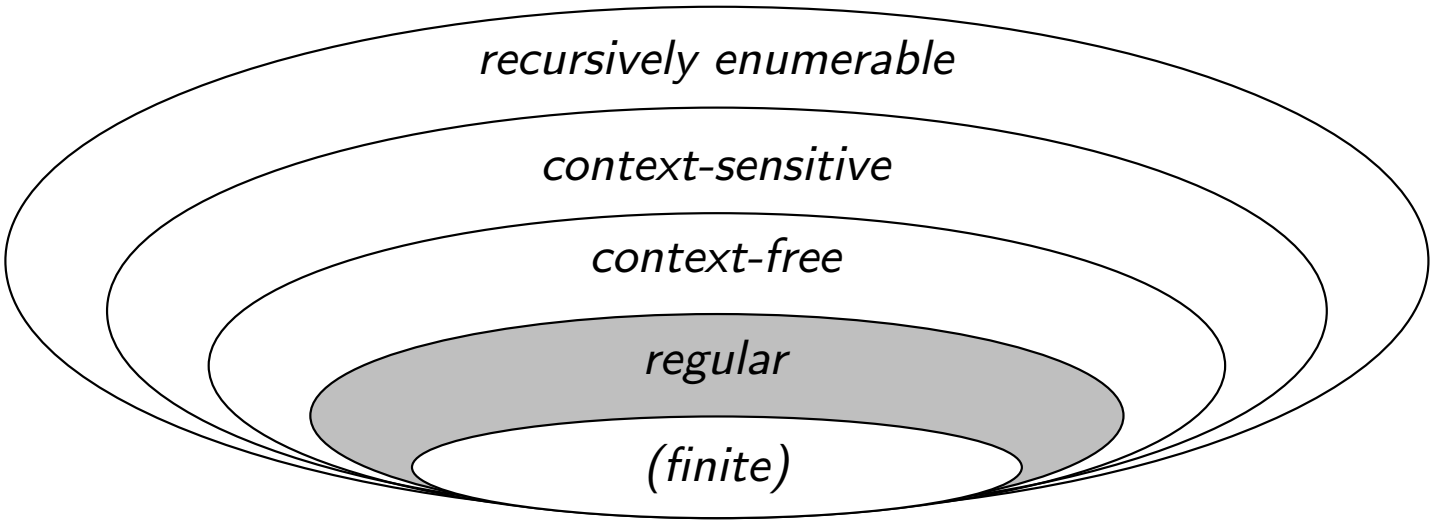
The formal class of subregular languages is where they meet.

Subregular languages

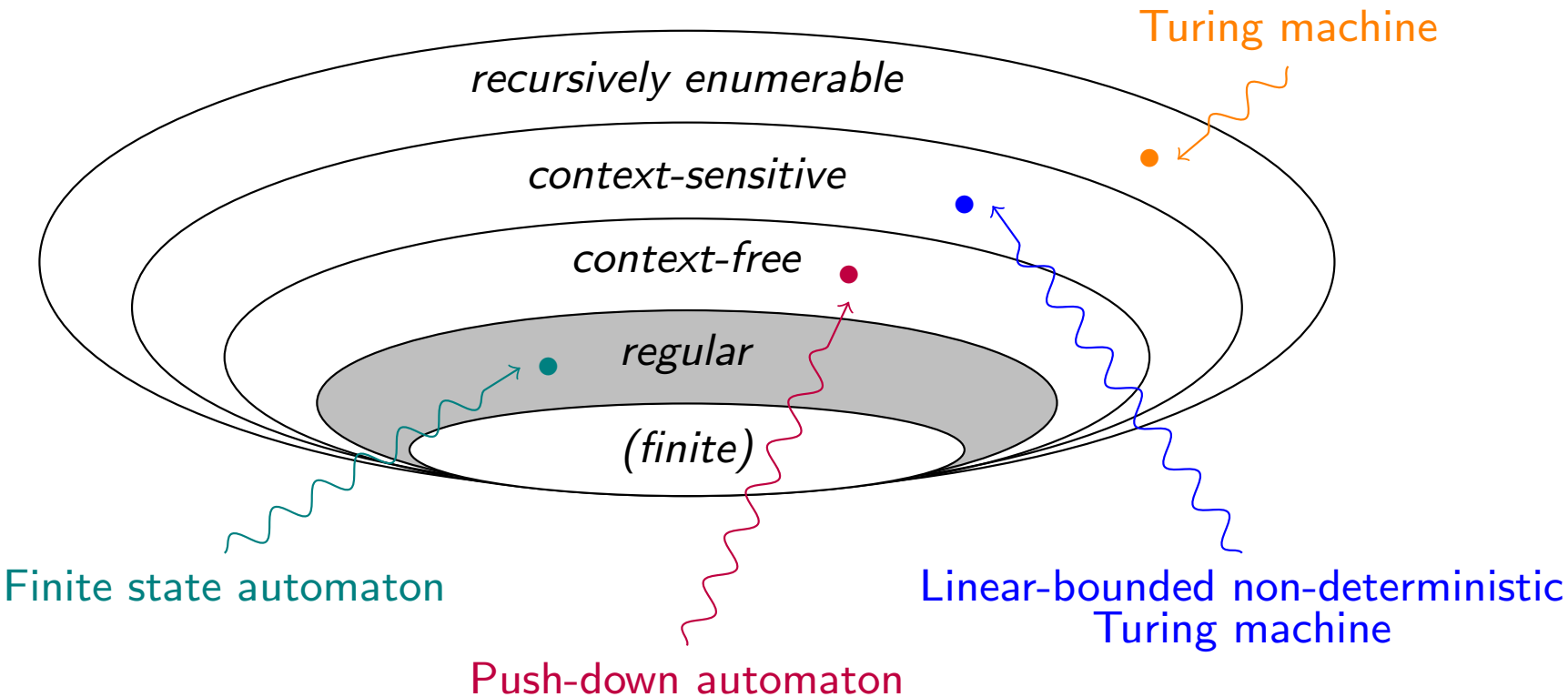
The formal class of subregular languages is where they meet.

- FLT: we know properties of subregular languages.
- Linguistics: and we know how to apply them!

The Chomsky Hierarchy of String Languages



The Chomsky Hierarchy of String Languages



Regular languages

Regular languages = FSA recognized = MSO definable

Regular languages

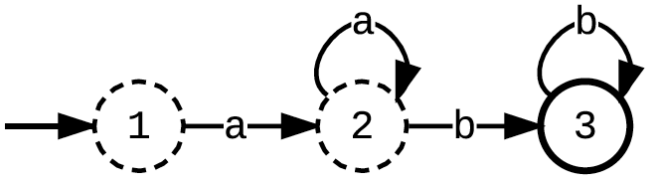
Regular languages = FSA recognized = MSO definable

a^+b^+

Regular languages

Regular languages = FSA recognized = MSO definable

a^+b^+



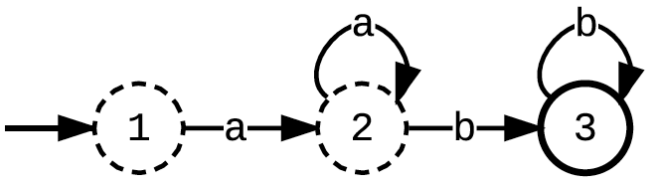
Regular

Regular languages

Regular languages = FSA recognized = MSO definable

a^+b^+

a^nb^n

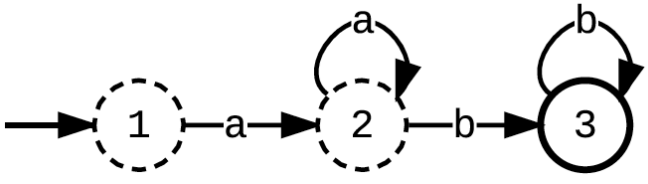


Regular

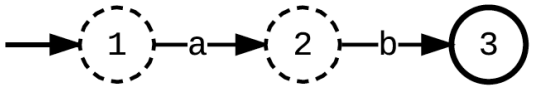
Regular languages

Regular languages = FSA recognized = MSO definable

a^+b^+



a^nb^n

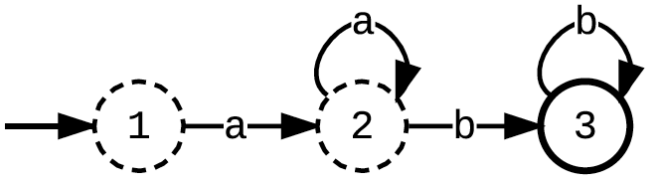


Regular

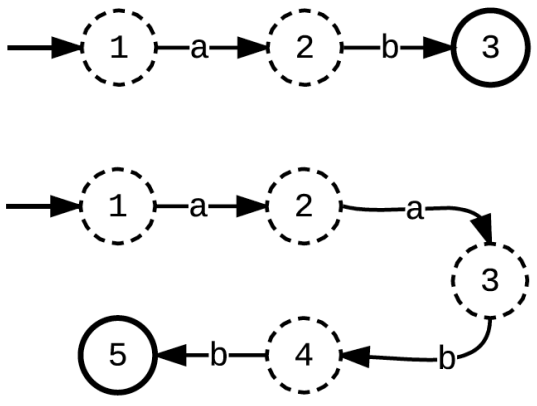
Regular languages

Regular languages = FSA recognized = MSO definable

a^+b^+



a^nb^n

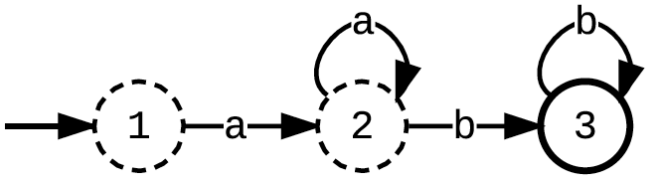


Regular

Regular languages

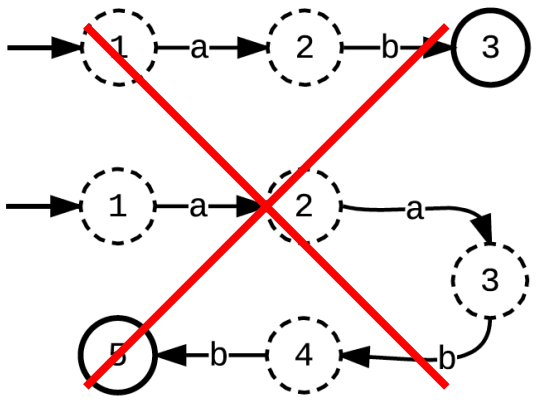
Regular languages = FSA recognized = MSO definable

a^+b^+



Regular

a^nb^n



. . . no general FSA can be constructed

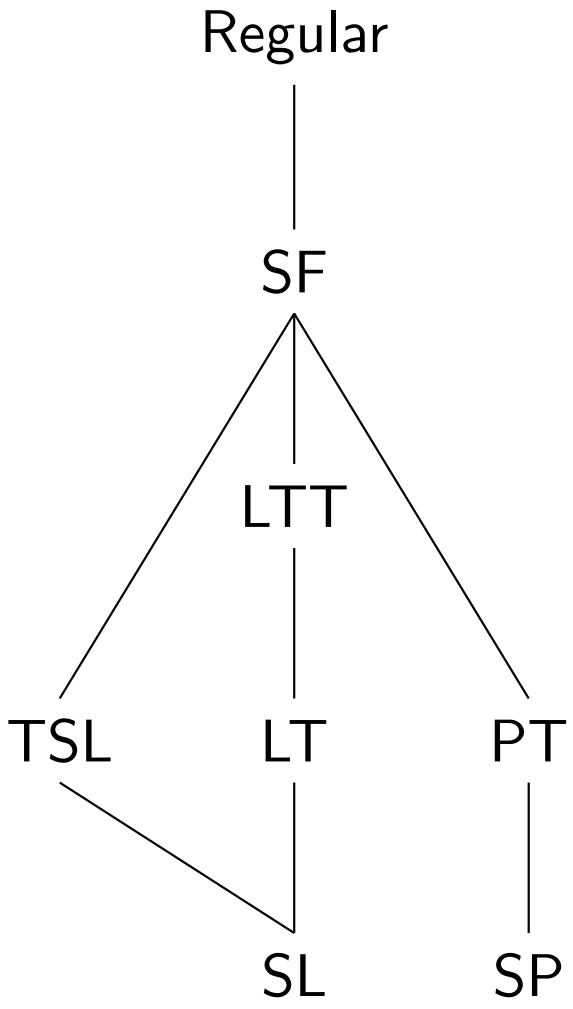
Context-Free

Subregular Hierarchy

The class of regular languages can be decomposed into **subregular hierarchy**

- Introduced by McNaughton&Papert (1971)
- Expanded by numerous researchers

Subregular hierarchy



Subregular Hierarchy

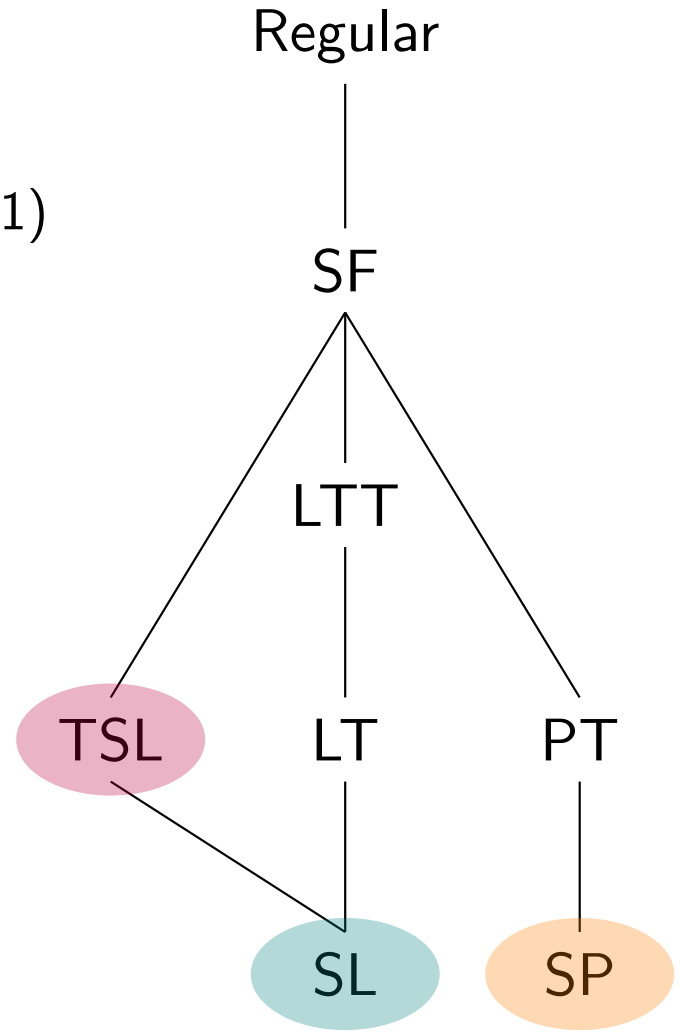
The class of regular languages can be decomposed into **subregular hierarchy**

- Introduced by McNaughton&Papert (1971)
- Expanded by numerous researchers

The most fruitful classes for the NL are:

- strictly local languages (SL)
- tier-based strictly local languages (TSL)
- strictly piecewise languages (SP)

Subregular hierarchy



(T)SL intuitions

Intuition: TSL is a *n*-gram model on steroids.

(T)SL intuitions

Intuition: TSL is a *n*-gram model on steroids.

n-gram model or SL grammar lists the (im)possible sequences of elements.

*ab constraint:

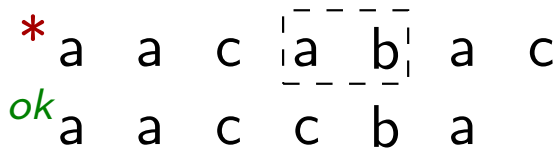
*	a	a	c	a	b	a	c
ok	a	a	c	c	b	a	

(T)SL intuitions

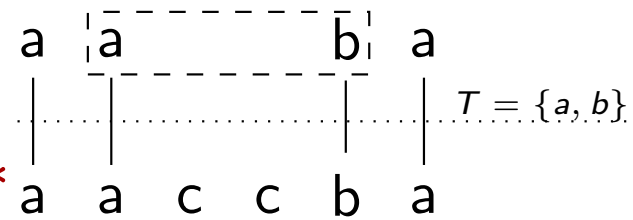
Intuition: TSL is a *n*-gram model on steroids.

n-gram model or SL grammar lists the (im)possible sequences of elements.

*ab constraint:



TSL grammar is a SL grammar of a certain subset of the alphabet (*tier alphabet*). Other elements are ignored.



Possible compounding patterns

Turkish

1. kapı
   ‘gate’
2. bahçe kapı-sı
   ‘garden gate’
3. türk bahçe kapı-sı
   ‘Turkish garden gate’

Possible compounding patterns

Turkish

1. kapı
   ‘gate’
2. bahçe kapı-sı
   ‘garden gate’
3. türk bahçe kapı-sı
   ‘Turkish garden gate’

word-(word^{+}-sı)

Possible compounding patterns

Turkish

1. kapı
   ‘gate’
2. bahçe kapı-sı
   ‘garden gate’
3. türk bahçe kapı-sı
   ‘Turkish garden gate’

word-(word^{+}-sı)

Russian

1. par
   ‘steam’
2. par-o-khod
   ‘steam boat’
3. par-o-khod-o-voz
   ‘steam boat carrier’

Possible compounding patterns

Turkish

1. kapı
   ‘gate’
2. bahçe kapı-sı
   ‘garden gate’
3. türk bahçe kapı-sı
   ‘Turkish garden gate’

word-(word^{+}-sı)

Russian

1. par
   ‘steam’
2. par-o-khod
   ‘steam boat’
3. par-o-khod-o-voz
   ‘steam boat carrier’

(word-o)*-word

Possible compounding patterns

Turkish

1. kapı
   'gate'
2. bahçe kapı-sı
   'garden gate'
3. türk bahçe kapı-sı
   'Turkish garden gate'

word-(word^{+}-sı)

Russian

1. par
   'steam'
2. par-o-khod
   'steam boat'
3. par-o-khod-o-voz
   'steam boat carrier'

(word-o)*-word

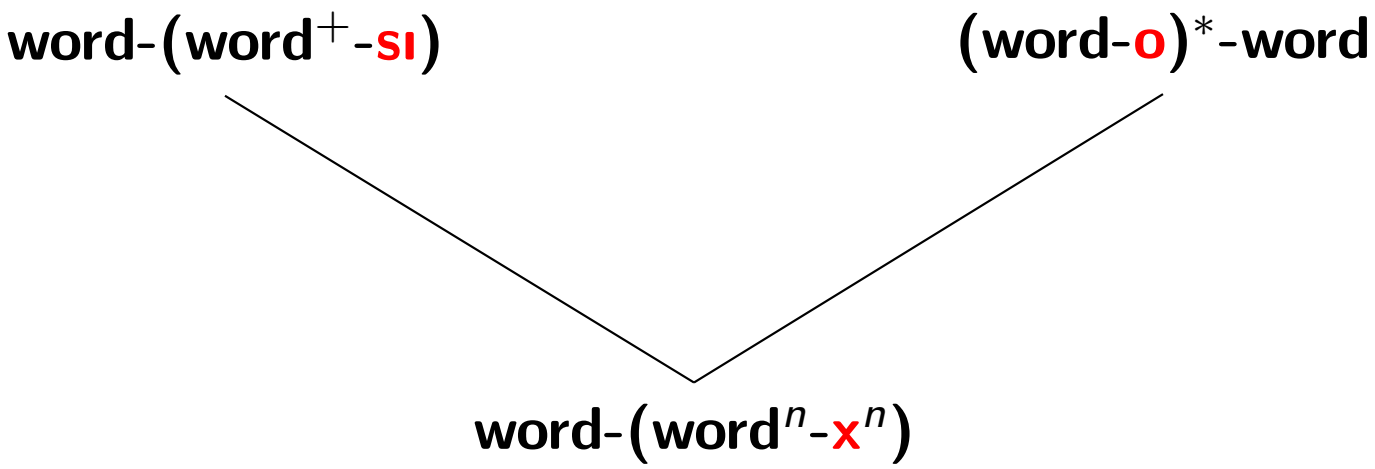
Both patterns are SL!

Impossible compounding pattern

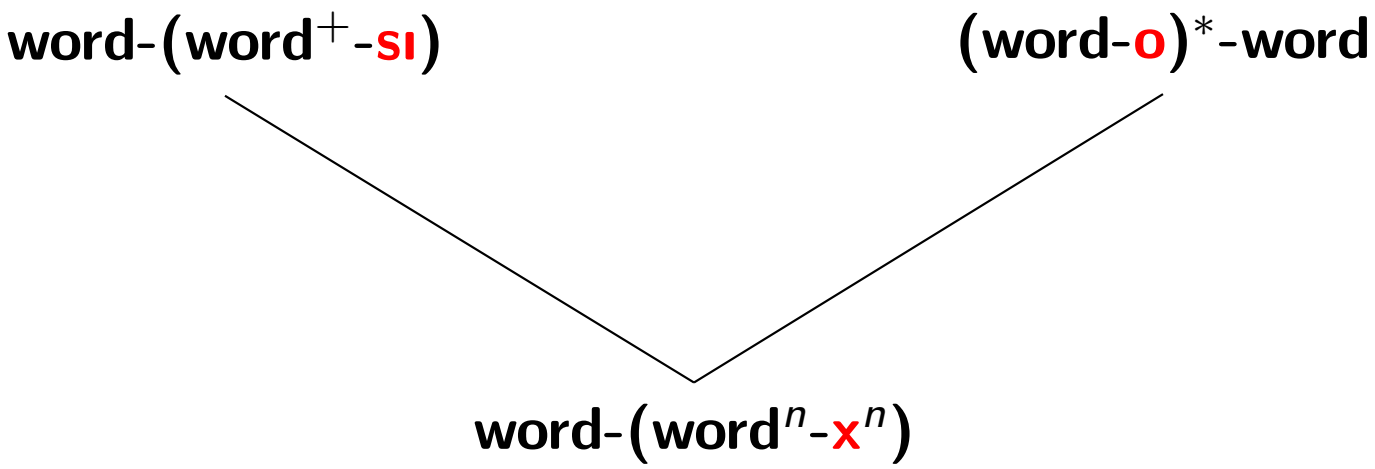
word-(word^{+}-sı)

(word-o)*-word

Impossible compounding pattern

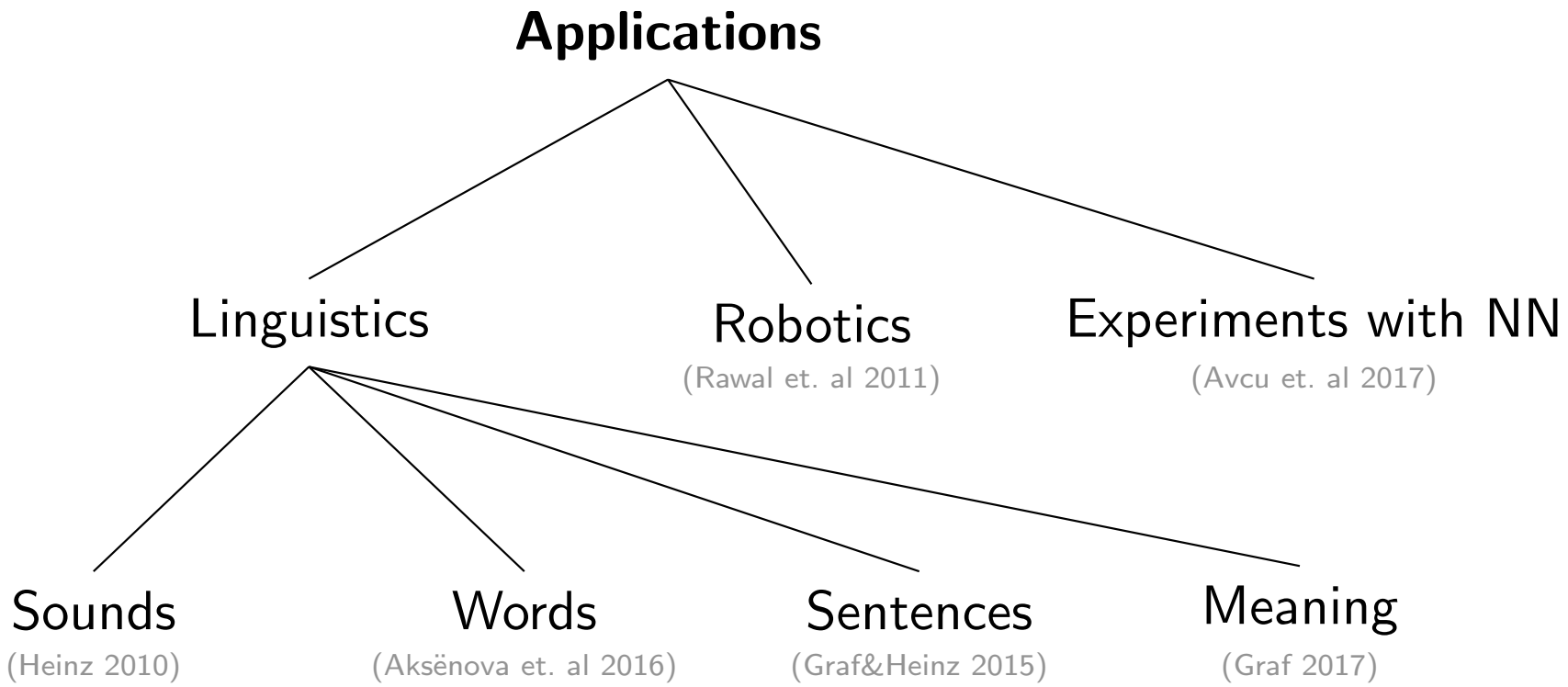


Impossible compounding pattern



This pattern is unattested!
Complexity: context-free.

What are the other applications?



Subregular toolkit: general info

`kist`: **k**ist **i**mplementing **s**ubregular **t**oolkit

Motivation: help researchers to avoid manual burden by providing a toolkit that contains necessary subregular functions.

- Python 3 (will be available via `pip`)
- Open source
- Available on GitHub
  `# https://github.com/loisetoil/slp`

What is available already?

Languages (positive and negative versions):

- Strictly local
- Tier-based strictly local
- Strictly piecewise

What is available already?

Languages (positive and negative versions):

- Strictly local
- Tier-based strictly local
- Strictly piecewise

Tools available:

- `learn`
  extracts the grammar
- `generate_sample`
  generates a data sample for a given grammar
- `scan`
  tells whether a string belongs to a language or not
- `fsmize`
  generates a FSA that corresponds to the grammar
- `generate_polarity`
  changes the polarity of the grammar

Subregular toolkit: examples

Initialization (positive SL)

```
>>> a = PosSL()
>>> a.data = ['abab', 'ab']
>>> a.k = 2
```

Subregular toolkit: examples

Initialization (positive SL)

```
>>> a = PosSL()
>>> a.data = ['abab', 'ab']
>>> a.k = 2
```

Learning (positive SL)

```
>>> a.learn()
>>> a.grammar
    [('>', 'a'), ('b', '<'), ('a', 'b'),
     ('b', 'a')]
```

Subregular toolkit: examples [cont.]

Scanning (positive SL)

```
>>> a.scan('ababab')
    True
>>> a.scan('aba')
    False
```

Subregular toolkit: examples [cont.]

Scanning (positive SL)

```
>>> a.scan('ababab')
    True
>>> a.scan('aba')
    False
```

Changing polarity (positive → negative SL)

```
>>> a.change_polarity()
>>> a.grammar
    [('a', 'a'), ('b', 'b'), ('>', '<'),
     ('a', '<'), ('>', 'b')]
```

Subregular toolkit: examples [cont.]

Learning (negative TSL)

```
>>> b = NegTSL()
>>> b.data = ['abaa', 'aab', 'ba', 'b']
>>> b.learn()
>>> b.grammar
    [('b', 'b'), ('>', '<')]
>>> b.tier
    ['b']
```

Subregular toolkit: examples [cont.]

Learning (negative TSL)

```
>>> b = NegTSL()
>>> b.data = ['abaa', 'aab', 'ba', 'b']
>>> b.learn()
>>> b.grammar
    [('b', 'b'), ('>', '<')]
>>> b.tier
    ['b']
```

Generating sample (negative TSL)

```
>>> b.generate_sample(3)
>>> b.data_sample
    ['abaaa', 'b', 'aaba']
```

What is currently happening?

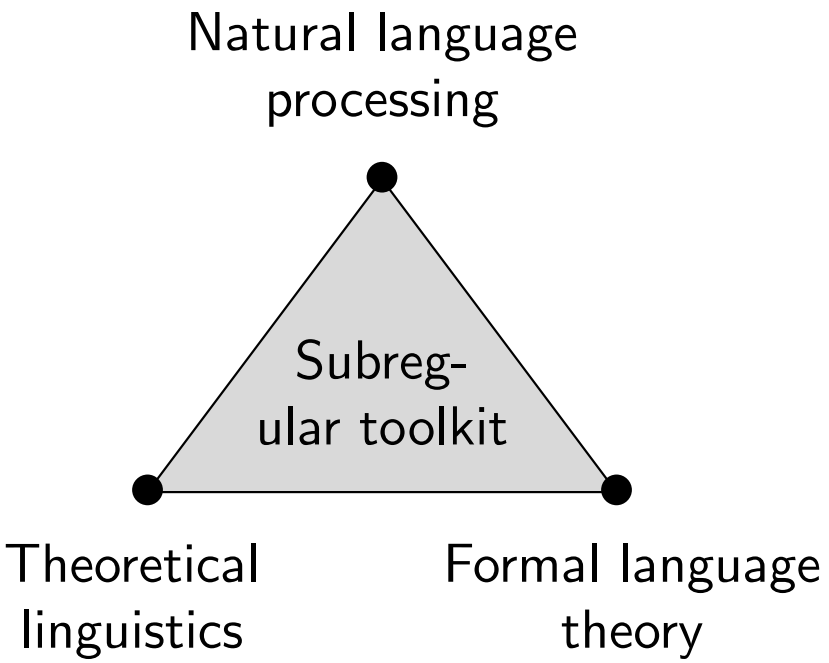
Currently being implemented:

- Probabilistic versions of SL, TSL, SP
- Subregular transductions
  - the grammars discussed above behave as *acceptors*: they either accept the given string or not;
  - *transducers* re-write the given string while reading it.

Summary

This subregular toolkit allows:

- test ideas currently available in literature;
- explore new methods to model NL;
- seek out new ways to improve the results.



Thank you!

> What I cannot create, I do not understand.
>
> *Richard Feynman*

References

Aksënova, Alëna, Thomas Graf and Sedigheh Moradi (2016)

Morphotactics as Tier-Based Strictly Local Dependencies.
In *Proceedings of SIGMorPhon 2016.*

Avcu, Enes, Chihiro Shibata and Jeffrey Heinz (2017)

Subregular complexity and deep learning.
In *Proceedings of the Conference on Logic and Machine Learning in Natural Language.*

Graf, Thomas and Jeffrey Heinz (2015)

Commonality in Disparity: The Computational View of Syntax and Phonology.
Slides of a talk given at GLOW 2015. Paris, France.

Heinz, Jeffrey (2010)

Learning long-distance phonotactics.
Linguistic Inquiry 41(4): 623 – 661.

Graf, Thomas (2017)

The subregular complexity of monomorphemic quantifiers.
Ms., Stony Brook University.

McNaughton, Robert and Seymour Papert (1971)

Counter-Free Automata.
MIT Press, Cambridge.

Rawal, Chetan, Herbert Tanner and Jeffrey Heinz (2011)

(Sub)regular Robotic Languages.
In *Proceedings of IEEE Mediterranean Conference on Control and Automation*, 321–326.